UNIVERSITY OF CALIFORNIA
Santa Barbara

Resource-Efficient Wireless Systems for Emerging Wireless Networks

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Lara B. Deek

Committee in Charge:

Professor Kevin C. Almeroth, Co-Chair

Professor Elizabeth M. Belding, Co-Chair

Professor Heather Zheng

September 2014

The Dissertation of
Lara B. Deek is approved:

Professor Heather Zheng

Professor Elizabeth M. Belding, Co-Chair

Professor Kevin C. Almeroth, Co-Chair

August 2014

Acknowledgements

I would like to dedicate this dissertation first and foremost to my family. Everything is a lot simpler when you have a family to support you and love you unconditionally. To my brilliant sister Joanna, my fondest memories at UCSB was waiting for you underneath the MRL after a day of work. To my kick-ass brother Ramzi, I have loved to watch you grow, enjoy SB as I have, and become the talented man you are today. Mom and Dad, no kid could ask for better parents. To my wonderful family, I am very lucky, very proud, and I would not be where I am if it were not for your love and support.

My extended family and close friends overseas as well. To Giorgio, full of critiques, fruitful discussions, and cultural and social understandings, and more importantly a full stock of anime dvd's. To George, my twin soul almost, who I can always count on to be there, even when you are in your deep sleep cycle, you would wake up. To Karl, my oldest friend since school, there's no filter around each other, we are goofy, vulgar, and just us.

To my friends from SB, the named and unnamed. The lasting relationships I have created with you, and how we have shared together whatever emotion that came to heart or mind (willingly or not). To Rahau, for being a harbor of zen and good energies and coconut oil. To Diana, my partner in crime and labored data structure classes. To Stephanie, my choir for healthy eating, exercising, and endorphins. To

Sumit, for being a calming and mindful spirit, detailed and encompassing conversations, and an inspiring mentor. To Janet, for our engaging discussions, fun girl times, and her unwavering support. To Sahar, the performance artist who feels close to my heart and soul. To Ludo, the climb-bing partner I explore whatever extreme activities with and last-minute plans. To Greg and Ben, pour nos conversations profondes et nos attention à [certains] détails. To Clemens, our passionate discussions, and to Manu, who would ask that Clemens and I take a break. To Alex, the sweetest bear hugs, most practical advice, and poking jokes. To Ali, the dolphins, buffaloes, and sense of humor, and Yanick who picked up the lantern and my losing $5 bet. To Adam, one of my first friends at UCSB and my first impression of it. To Gianluca and our silly moments and for inventing *Deek'ing out*. To my French Press pals, James, Taylor, Brian, and Justin for keeping me company while I work and jamming the music after-hours.

To the people I have worked with professionally, collaborated with, and been advised by: Kevin Almeroth, Elizabeth Belding, Heather Zheng, Sung-Ju Lee, Sumit Singh, Sharad Agarwal, Ben Zhao, Ram Seshadri, Janet Kayfetz, Eduard Garcia-Villegas, Mike Wittie, and Xia Zhou. You have helped me grow professionally, and I am sure I have the foundation to continue doing so.

And last but not least, to the beautiful and wonderful city of Santa Barbara, you have simply spoiled me. I have picked up a California lifestyle here, and I do not intend to change it.

As Gibran Khalil Gibran says, “Life without love is like a tree without blossoms or fruit.” Santa Barbara, my life here has been a long spring.

Curriculum Vitæ

Lara B. Deek

EDUCATION

Certificate in College and University Teaching *June 2014*

University of California, Santa Barbara Santa Barbara, CA

B.E. Computer and Communications Engineering *June 2008*

(*Dean's Honor List*)

American University of Beirut Beirut, Lebanon

CONFERENCE AND JOURNAL PAPERS

1. Utkarsh Goel, Lara Deek, Mike P. Wittie, and Qing Yang. "Characterizing Challenges to End-to-End Application Performance and User QoE on Mobile Devices." In *IEEE Communications Surveys and Tutorials (to be submitted)*, August 2014.
2. Lara Deek, Eduard Garcia-Villegas, Elizabeth Belding, Sung-Ju Lee, and Kevin Almeroth. "Joint rate and channel width adaptation in 802.11 MIMO wireless networks." In *IEEE Secon*, June 2013.

   Nominee for the Best Paper Award.

3. Lara Deek, Eduard Garcia-Villegas, Elizabeth Belding, Sung-Ju Lee, and Kevin Almeroth. "Agile rate and channel width adaptation for MIMO WLANs." *IEEE Transactions on Mobile Computing (under submission)*, 2013.

4. Lara Deek, Eduard Garcia-Villegas, Elizabeth Belding, Sung-Ju Lee, and Kevin Almeroth. "Intelligent channel bonding in 802.11n WLANs." *IEEE Transactions on Mobile Computing*, 13(6):1242-1255, June 2014.

   TMC Spotlight Editorial: Highlight in IEEE Computer.

5. Lara Deek, Eduard Garcia-Villegas, Elizabeth Belding, Sung-Ju Lee, and Kevin Almeroth. "The impact of channel bonding on 802.11n network management." In *ACM CoNEXT*, Dec. 2011.

6. Lara Deek, Xia Zhou, Kevin Almeroth, and Haitao Zheng. "To preempt or not: Tackling bid and time-based cheating in online spectrum auctions." In *IEEE Infocom*, Apr. 2011.

7. Mike P. Wittie, Veljko Pejovic, Lara Deek, Kevin Almeroth, and Ben Zhao. "Exploiting locality of interest in online social networks." In *ACM CoNEXT*, Nov. 2010.

8. Khaled Harras, Lara Deek, Caitlin Holman, and Kevin Almeroth. "DBS-IC: An adaptive data bundling system for intermittent connectivity." *SigComm Computer Communications Review*, 32(16):1687–1698, Oct. 2009.

9. Lara Deek, Sara Thoubian, Serouj Jamijian, Khaled Harras, and Hassan Artail. "Exploiting parallel networks in intermittently-connected mobile environments." In *IEEE Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2008.

10. Lara Deek, Kevin Almeroth, Mike P. Wittie, and Khaled Harras. "Exploiting parallel networks using dynamic channel scheduling." In *Wireless Internet Conference (WiCon)*, Nov. 2008.

Abstract

Resource-Efficient Wireless Systems for Emerging Wireless Networks

Lara B. Deek

As the wireless medium has become the primary source of communication and Internet connectivity, and as devices and wireless technologies become more sophisticated and capable, there has been a surge in the capacity demands and complexity of applications that run over these wireless devices. To sustain the volume and QoE guarantees of the data generated, the opportunity and need to rethink wireless network design across all the layers of the protocol stack has firmly emerged as a solution to enable the timely and reliable delivery of data, while handling the inherent challenges of a crowded wireless medium, such as congestion, interference, and hidden terminals.

The research work presented in this dissertation builds efficient solutions and protocols with a theoretical foundation to address the challenges that arise in rethinking wireless network design. Example challenges include managing the overhead associated with complex systems. My work particularly focuses on the opportunities and challenges of sophisticated technology and systems in emerging wireless networks. I target the main thrusts in the evolution of wireless networks that create

significant opportunity to achieve higher theoretical capacity, and have direct implications on our day-to-day wireless interactions: from enabling multifold increase in capacity in wireless physical links, to developing medium access techniques to exploit the high speed links, and making the applications more bandwidth efficient. I build deployable, and resource-aware wireless systems that exploit higher bandwidths by leveraging and advancing diverse research areas such as theory, analysis, protocol design, and wireless networking. Specifically, I identify the erroneous assumptions and fundamental limitations of existing solutions in capturing the true and complex interactions between wireless devices and protocols. I use these insights to guide practical and efficient protocol design, followed by thorough analysis and evaluation in testbed implementations via prototypes and measurements. I show that my proposed solutions achieve significant performance gains, at minimum cost to overhead.

Contents

Chapter 1

Introduction

1.1 Motivation and Overview

The proliferation of portable wireless devices and the widespread adoption of IEEE 802.11-based WLANs has lead to the wireless medium becoming the dominant method of communication and Internet connectivity. According to a CISCO report, the number of portable wireless devices outnumbered the human population in 2013, and this growth shows no indication of slowing down [74]. As these devices and wireless technologies become more sophisticated and capable, there has been a surge in the capacity demands of the applications that run over these wireless devices. Applications that once ran on a wired connection are now expected to run seamlessly on a wireless connection. From portable devices alone, traffic

grew 70% in 2013, nearly double what it was the year before. The mobile commission itself estimates that mobile broadband traffic will increase more than thirty fold by 2015 [70, 114]. It is clear that wireless mobile data traffic and the demand for bandwidth is growing, and at an impressive rate.

While we continue to witness a surge in mobile data traffic, the usage of mobile Internet introduces challenges to application performance that are intrinsic to this shared wireless medium. These challenges include congestion, interference, intermittent connectivity, and hidden terminals. My conjecture in this PhD dissertation is that, in order to sustain the increase in volume and complexity as well as the QoE guarantees of mobile data traffic growth, while handling the inherent challenges of a crowded wireless medium, there is a need to rethink wireless network design across all layers of the protocol stack, as a solution to enable the timely and reliable delivery of data.

In this dissertation, I recognize and discuss directions that have given me the opportunity to design novel and practical, or deployable, solutions to rethinking wireless network design. I build efficient solutions and protocols that address the complexity challenges that arise in these wireless systems. I particularly focus on the opportunities and challenges enabled through novel, or emerging, and sophisticated technology and systems in wireless networks. Further, as we continue to witness innovations in mobile applications, and in order to support their increasing

capacity demands, I specifically target the main thrusts in the evolution of wireless networks and technologies that create significant opportunity to exploit higher theoretical capacity.

In the process of identifying the relevant research directions, I have been inspired by two fundamentally essential and complementary themes: (1) how to increase wireless network capacity, and (2) how to more efficiently utilize the capacity already available. I address these themes in my research from two perspectives. The first perspective seeks to achieve the specific QoE requirements of upper-layer user applications, by *making applications more bandwidth efficient*. The second perspective seeks to achieve the full capacity performance of sophisticated lower-layer wireless technology, by *developing medium access techniques to exploit high speed links*, or *by enabling multifold increases in capacity in wireless physical links*. I believe that these two perspectives represent the flip-side of the same coin: a top-down and bottom-up approach to redesigning network protocols. Both approaches together enable the design of practical and deployable solutions to ensure that we meet the ever-growing demand for wireless network resources and the QoE guarantees of the data generated. These perspectives also effectively distribute performance responsibilities to the appropriate layers in the networking stack. Within these themes, there are four primary directions for contribution that I identify in my research, as

depicted in Figure 1.1. These directions together contribute to the design of efficient and high-capacity wireless systems.

My first research direction is inspired by the need to meet QoE requirements of upper-layer applications. With the prevalent usage and increasing capability of mobile devices, there has been a rapid shift of user applications towards mobile devices. A majority of the modern and emerging applications on these devices requires Internet access for full and even effective operation, either to upload crucial information, or to simply display content. At the same time however, these applications that are designed with the assumption of a single constant and reliable connection to the Internet, now have to operate in environments characterized by intermittent connectivity and fading signal quality. The application might be running on a wireless connection that fades due to limited range or congestion overload. This fade will lead to incomplete and thus unviewable data, and sometimes even full application failure. Furthermore, previously viewed data is often not cached and so is lost in the absence of an Internet connection. The challenges faced in mobile environments motivates the design of application-aware wireless protocols. These protocols will allow applications to operate more seamlessly in such environments and to take advantage of high bandwidth when available.

A tangent opportunity to increase capacity in mobile environments is to exploit the multiplicity of network technologies integrated in modern-day wireless portable

devices. With the proliferation of wireless technologies and network deployments, it is not uncommon to come across multiple Internet access opportunities. Users can now achieve higher capacity from utilizing multiple co-existing networks. Applications however are not designed to exploit the opportunities and mitigate the challenges of multiple wireless technologies and their characteristics. Therefore, the second research direction is the design of solutions that account for and maximize on the differences in wireless connection characteristics between technologies, namely differences in data rate, delay, availability, and cost.

The more obvious challenge that is created by the continuing growth of wireless traffic is the increase in load on limited spectrum. Due to the inherent broadcast nature of the wireless medium, an increase in traffic can lead to congestion, loss, and, consequently, degradation in application performance. With the FCC opening white space spectrum in 2009 for unlicensed use, wireless networks that once operated on crowded spectrum, now had the flexibility to gain access to and operate on a wider range of spectrum. The challenge however is to design spectrum access models that ensure spectrum efficiency and social welfare (i.e. fairness between users) in this open white space spectrum. The third direction in my research is in identifying techniques to efficiently and fairly distribute or allocate bandwidth in the white space spectrum between participating users.

Another marked improvement in wireless network capacity has occurred with the emergence of the IEEE 802.11n and now 802.11ac WiFi standards. Through the addition of a number of new technologies to the WiFi chipset, more prominently through sophisticated smart-antenna technology and the support of wider channel widths, the WiFi supported data rates have increased exponentially, and have now reached wired Ethernet speeds with 802.11ac. Applying or exploiting these added 802.11 next-generation, high-bandwidth technologies however, is a challenge, and it is only through the mindful application and intelligent usage of these technologies that we can exploit their bandwidth opportunities. Therefore, the fourth direction in my research is in the design of wireless systems that can exploit these sophisticated technology and their promised data rates.

The four main research directions I identify and tackle in this dissertation have direct implications on our day-to-day interactions and make significant contributions towards meeting the ever-growing demand for wireless network resources and performance. I build much needed practical, deployable, and resource-aware wireless systems that exploit higher bandwidths and configure wireless parameters to accommodate traffic demands. The design of these systems is achieved by leveraging and advancing diverse research areas such as theory, analysis, protocol design, and wireless networking. Specifically, I identify the erroneous assumptions and fundamental limitations of existing solutions in capturing the true and complex interactions be-

tween wireless devices and protocols. I use these insights to guide practical and efficient protocol design, followed by thorough analysis and evaluation in testbed implementations via prototypes and measurements. The system approach to the design of resource-efficient wireless systems can be summarized and described as an *observe-infer-exploit* methodology (i.e. as explained in Figure 1.2).

1.2 Thesis Statement

As wireless network deployment and usage evolves and increases in volume and complexity, we are required to rethink wireless network design across all the layers of the protocol stack: from enabling multifold increase in capacity in wireless physical links, to developing medium access techniques to exploit the high speed links, and making the applications more bandwidth efficient.

This dissertation proposes novel and practical solutions for each of these subproblems with a goal of realizing these solutions over current and emerging wireless networks.

1.3 Dissertation Organization

This dissertation is organized into six chapters. Each chapter exposes and discusses a direction in current and emerging wireless networks that has presented an opportunity to exploit novel and practical solutions that address the evolution and increase in volume and complexity of wireless network deployment and usage. Particularly, the chapters address each of the three sub-problems presented in this dissertation's thesis statement to rethinking wireless network design, namely: (1) enabling multi-fold increases in capacity in wireless physical links, (2) developing medium access techniques to exploit high speed links, and (3) making the applications more bandwidth efficient.

Figure 1.1 depicts the four main emerging directions and contributions throughout the six chapters of this dissertation. Each direction in Figure 1.1 has presented an opportunity to move to higher bandwidths in wireless networks. However, in order to exploit those bandwidth opportunities, each direction has presented a unique set of challenges that I have had to tackle in my work (cf. Chapter 1.1 for an overview of the challenges). I discuss these challenges in great detail within each chapter, and finally, Figure 1.2 summarizes the contributions of each individual chapter as well as the logical flow of inquiry in the solution design.

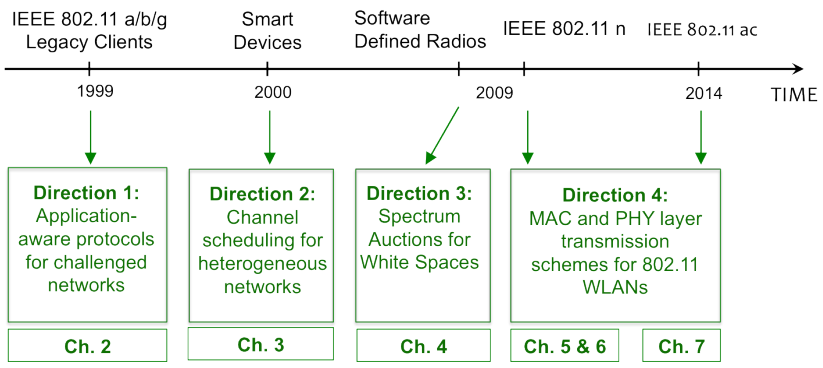


Figure 1.1: The four main directions to exploit bandwidth opportunities and complex systems in emerging wireless networks. The chapters indicate this dissertation's contributions towards each direction.

Chapter 2 presents our design of an application-aware protocol that mitigates the challenges of mobile networks on application performance. This is done by exploiting periods of free and fast connectivity to prefetch and cache content that will otherwise be viewed by the user, thus *making the applications more bandwidth efficient*. In this chapter, we solve the more complicated problem of complete app disconnectivity, thus solving the problem of avoiding slow and expensive wireless networks. There are times when a phone or tablet is not connected to the Internet – either because there is no useable wireless signal, or the network is too expensive for the user such as when roaming internationally. When apps are launched during this disconnection, many will fail with an error message to the user, or show stale content from the last time the user launched the app. My solution, *Airplane Mode*, gives the user the illusion of connectivity in mobile applications while disconnected.

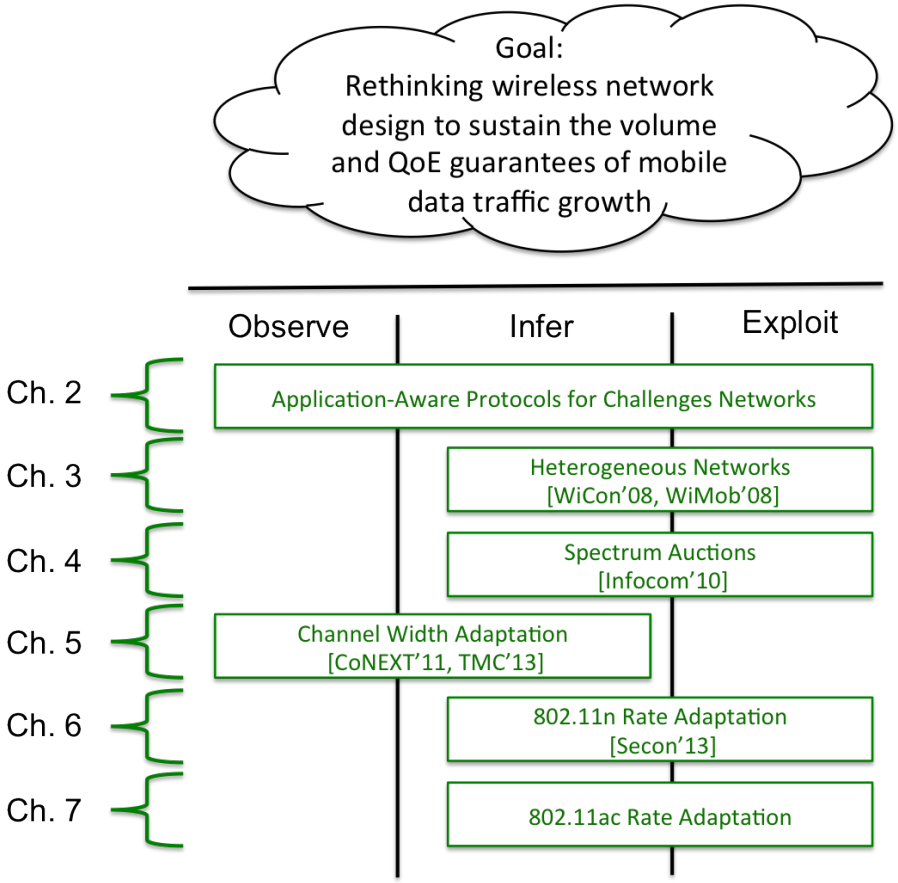


Figure 1.2: Dissertation overview with individual chapters and their corresponding projects categorized with respect to the stages of the methodology flow: *observe-infer-exploit*.

The user sees content from the last time the phone or tablet was connected to the Internet, not the last time the app was launched. *Airplane Mode* passively observes app behavior during normal operation, builds a machine learning model over time of what web content the app accesses, prefetches that content when asked to, and when disconnected mimics Internet connectivity to apps while intercepting outgoing requests and responding from that prefetched content.

With the proliferation of wireless and mobile devices, we now have multiple connection opportunities available to a single device. Chapter 3 proposes exploiting the bandwidth opportunities provided by these added technologies, or interfaces, so as to enhance user experience and data rate, thus *enabling multifold increases in capacity in wireless physical link*. Using the characteristics of the wireless technologies, namely cost, delay, and data rate, I identify the cost factor that corresponds to different achieved performance data rates. These models serve as input to a system that can dynamically choose the best operating point based on a performance to cost tradeoff.

In Chapter 4, we develop an efficient and fair solution to distribute or allocate bandwidth in the white space spectrum. By efficiently exploiting uncrowded frequency, white space spectrum presented an opportunity to *enable multifold increases in capacity in wireless physical links*. More specifically, we adopt an auction mechanism for white spaces and develop an online spectrum auction which we call *Topaz*.

Topaz accepts bidder requests in real time and schedules them in such a way as to achieve spectrum efficiency and social welfare. We also introduce a preemptive model to our spectrum auction design, which can preempt (or remove) a bidder from the auction based on newly arriving bidder requests. This preemption mechanism allows us to control the observed tradeoff between spectrum efficiency and social welfare in online spectrum auctions. We introduce a scale that allows us to control the extent of bidder preemption so as to enourage both bidder and auctioneer participation.

In Chapter 5 through the last Chapter 7, I study, develop, and propose methods of exploiting the exponential increases in data rate enabled through added WiFi technologies, namely wider channel widths and higher-order MIMO smart-antenna technologies. More specifically, I *develop medium access techniques to exploit the high speed links* enabled through sophisticated WiFi technologies.

Chapter 5 describes my work on channel width performance in IEEE 802.11 WiFi networks. With the emergence of IEEE 802.11n and 802.11ac, there is new support for the utilization of wider bandwidths. The 802.11n standard supports channel bonding two 20MHz channels into a single 40MHz channel. The 802.11ac standard supports not only 20MHz and 40MHz bandwidths, but also 80MHz and 160MHz bandwidths, where a 160MHz channel can consist of two contiguous as well as two discontiguous 80MHz channels. By operating on wider bandwidths, we

can send more data and therefore increase the transmission data rate. Wider bandwidths however become more susceptible to interference. Using measurements and observations from real testbed deployments, I evaluate the performance of wider channels and identify the opportunities and usage-terms to exploit wider channel widths and intelligently incorporate wider channel width operation in network deployments to maximize performance and efficiency.

Chapter 6 extends the work on IEEE 802.11n WiFi resource management through the design of a rate adaptation solution. From our previous work on channel width adaptation, I find that channel width adaptation, referred to as channel management, should be done in conjunction with rate selection in order to maximize performance. Through the addition of Multiple-Input Multiple-Output (MIMO) antenna technology in 802.11n, rate selection also becomes a challenging problem. In my work, I develop a novel, standard-compliant, closed-loop rate adaptation solution which adapts both rate and bandwidth jointly. Then using a real testbed implementation, I show that such an approach achieves high accuracy and outperforms existing and popular 802.11n rate adaptation solutions. I call this solution *ARAMIS* (Agile Rate Adaptation for MIMO Systems).

Chapter 7 describes my work on applying Multi-User (MU) MIMO (Multiple-Input Multiple-Output) technology to WLANs, within the context of the 802.11ac standard. MU-MIMO can provide significant spectral efficiency gains by allowing

an AP to serve multiple clients simultaneously without requiring much added complexity or a larger form factor at the clients (unlike equivalent large-order single user (SU) MIMO techniques). Applying MU-MIMO to 802.11 WLANs, however, poses unique system constraints across the different layers of the protocol stack, from the physical layer signal processing and channel feedback to medium access and resource allocation. In my work, I have investigated and developed solutions towards the successful deployment of MU-MIMO systems in 802.11 environments. Specifically, I identify practical techniques for transmit precoding and receive signal processing, and formulate and address a critical OFDM MU-MIMO resource allocation and utilization problem, namely data rate selection. I demonstrate the efficacy of my proposed MU-MIMO framework components via evaluation that highlights the performance gains in terms of system utility metrics.

Ultimately, I believe that though this dissertation has tackled each of the directions summarized in Figure 1.1 separately, the directions must come together in order to present practical, sustainable, and scalable solutions that solve both the current and future growths in mobile data traffic as well the increase in application sophistication and satisfying the associated user QoE. I present this vision as a future research direction which I discuss in detail in Chapter 8.

1.4 Contributions

In this dissertation, I make several contributions towards the design and building of resource-efficient solutions and protocols to sustain the volume and QoE guarantees of mobile data traffic growth. In this section, I present the main contributions in this dissertation work, categorized based on the contribution to the overall system/cycle. The impacts of my work on the broader research community are:

- Measure and Analyze
  - Evaluate impact of emerging network architectures on application performance
  - Identify the factors that impact channel width in real WiFi settings
  - Evaluate practical link metrics to characterize MIMO performance
- Infer and Develop Models
  - Recommendation system to improve application performance in mobile environments
  - Design of wireless systems and architectures for mobile environments
  - Properties of scheduling system for heterogeneous networks
  - Properties of a truthful online spectrum auction

  - Properties of channel widths in WiFi networks
  - Identification of a practical link metric to characterize MIMO performance
  - Identification of practical and informed approaches to apply MU-MIMO to 802.11 OFDM, multi-carrier environments

- Design Resource-Efficient Wireless Systems
  - Application-friendly protocol for mobile environments
  - Dynamic scheduling system for heterogeneous networks
  - Tackling bid and time-based cheating in online spectrum auctions for white spaces
  - Novel approach to rate adaptation in 802.11n networks
  - Formal and practical definition and application of rate adaptation components in OFDM MU-MIMO WLANs

Chapter 2

Application-Aware Protocols for Mobile Networks: An Online Experience for Mobile Devices

2.1 Introduction

As the wireless medium has become the primary source of communication and Internet connectivity, and with the widespread proliferation of portable devices, there has been an increase in popularity of the online mobile app market. Applications however that are designed with an assumption of constant and high quality connectivity have to now operate in mobile environments with time-varying characteristics. Many mobile apps rely on network connectivity for key app functionality. This introduces new challenges to application QoE, such as intermittent connectivity and fading signal quality. These challenges lead to user frustration due to large latencies, incomplete and thus inaccessible data transfers, and possibly even com-

plete application failure. This work seeks to mitigate the effects of mobility on the performance and operability of existing applications by making applications more bandwidth efficient.

In addition to the variability of connectivity in mobile environments, a second challenge in current wireless network deployments is the variability of network bandwidth. It is not unlikely that we move between wireless networks with varying bandwidth and cost constraints. For example, a person might have fast and free connectivity at home and in the office, but expensive and slow connectivity in between. In this work, we want to give the user the experience of connectivity in apps when her mobile device is offline. We consider a device to be offline when it has no network connectivity, or has marginal performance due to poor signal strength or congestion, or is connected to an expensive network such as international cellular roaming. Our goal is for this experience to mimic the experience the user would have had if she launched the app when she was *last online*, not when she last launched the app which could have been hours or days ago. Solving the impact of mobility on application QoE for the extreme scenario of disconnectivity also solves it for the slow and expensive connectivity scenario.

In the design of an online experience for disconnected and offline apps, a naïve solution would periodically launch every app in the background, let it access the Internet, and then exit the app. Later, when the user launches an app while offline,

she may see content from when the device was last online. This is unsatisfactory for a number of reasons. Some apps do not cache network content and will throw an error message if launched while offline. Some apps do not fetch all web content on launch, but instead rely on the user to click on icons or other UI elements to initiate specific downloads. Finally, the overhead of launching every app that a user may have on her phone can have a drastic impact on the phone's battery and performance.

To solve this problem, we present the design and implementation of Airplane Mode. Our vision is that our system will replace the current "airplane mode" functionality on mobile devices. Today, this switch simply turns off all wireless interfaces. In our system, enabling "airplane mode" causes our system to fetch all the web content that apps may need while offline, subsequently trick those apps into believing they are online, and provide that fetched content to those apps when they request them as if it is coming from the Internet.

Prior work [66] addresses this problem for individual web pages in a mobile browser. Web pages are typically designed to be fetched by a variety of different browsers, by following web standards for embedding content. In this way, web browser behavior is relatively predictable. Apps do not follow such pre-established logic. An app may fetch an XML URL on launch, select some individual URLs listed in that XML file, modify those URLs, fetch them, parse those files, construct new URLs using strings in those files, and then fetch those. Apps are not constrained

by the rules of the browser, and have custom logic for identifying, fetching, and displaying web content to the user. As a result, solving this problem for arbitrary apps is fundamentally harder.

Airplane Mode passively observes what web content each app fetches during normal operation. It uses a set of generic filters to identify URLs and snippets of URLs and keywords in the XML, JSON, and other web content that the app fetches. Over time, it builds a machine learning model that ties each of these observed text to the actual URLs that the app fetches. Airplane Mode handles a variety of transformations that the app may apply on these URLs, such as replacing a .html suffix with .xml, or passing an image URL through a transcoding service that changes its size. When going offline, Airplane Mode will apply this learned model to fetch web content for the app, parse that content, identify additional URLs that the app may fetch, and fetch those as well. If the app is launched by the user while offline, Airplane Mode fakes the presence of an Internet connection, and intercepts and responds to web requests by the app.

Airplane Mode is ideally suited to apps that primarily display content to the user. News, social media, education, photo browsing, comic strip, and weather apps will work well, though will fail to allow the user to post transactions on the web services while offline (such as make a Facebook post). Some apps that are interactive by nature, such as multiplayer games and online banking, are not suitable.

Our primary contributions include the design and implementation of a practical and functional system that enables app operability when offline. Determining when to enable Airplane Mode is not in the scope of this chapter; we build a system where the user manually turns it on and off. Our system is implemented on Windows 8.1 tablets. We evaluate the accuracy of our system in prefetching and caching content as well as the associated system overhead. Our lab results demonstrate high prefetching and prediction success rate across 10 apps at a reasonable cost to system overhead in terms of total bytes downloaded.

2.2 Design considerations

Our design of Airplane Mode is heavily influenced by how apps access the Internet, and the goals we have on performance, applicability to different apps, and adapting to user behavior.

2.2.1 Challenges

To inform our design, we have studied how apps behave with respect to what web content they download. We have manually examined the MSIL code of several tens of highly rated Windows Phone 8 apps (similar to examining the Java byte code

of Android apps). We have also manually examined network traces of few tens of highly rated Windows Phone 8 and Windows 8.1 tablet apps [1].

We have found that apps typically behave very differently than web browsers. When a user enters or clicks on a URL in the browser, such as `http://www.cnn.com/`, a standard set of actions are taken by the browser. The browser will begin by fetching the default object at this URL, which may be `index.html`. That HTML page may refer to images, CSS, javascript and other URLs using standard HTML tags that the browser will fetch, execute, and render. In this way, browser behavior in visiting a website is not only predictable, it is similar between different browsers and can be determined without observing past browser behavior – one simply needs to fetch the starting URL and follow the same rules that any browser would follow. This is a direct result of web standards that are designed for interoperability across different browsers on different operating systems.

When an app, such as a news app or social networking app, is launched, it will typically fetch one or a few URLs that are deterministic. These URLs may be statically defined or hardcoded in the app's source code, and sometimes may include a user ID or device ID that can vary between different users. The underlying OS will fetch these URLs for the app and return the contents. The contents may be a variety of standard web objects, such as XML, XHTML, JSON, JPG, PNG, and un-

[1] In this work, we consider only "modern" apps that are purchased by users from the app store on Windows.

formatted text. Some of these objects, such as XML, may contain URLs or portions of URLs. The app will parse out these URLs and use some custom logic to fetch a subset of them. It may modify some of the URLs before fetching them. The contents of those fetches may in turn identify additional URLs that the app may fetch. Some of these fetches may be automatic as a result of launching the app, while some may happen only if the user clicks on an icon in the display. In this way, the app operates in a fetch – parse – compute – fetch cycle with respect to downloading web content.

In the rest of this chapter, we classify the set of URLs that an app may fetch into four different buckets: *static*, *modified*, *verbatim*, and *partial* URLs. *Static* URLs are those URLs that are either hardcoded in the source code of the app or calculated deterministically and do not change through multiple app executions. They are fetched when the app is launched, or when a major placeholder, such as the sports section of a news app, is clicked for viewing. Static URLs are typically not present in the contents of previous URL fetches. On the other hand, *modified*, *verbatim* and *partial* URLs can be observed in previous HTTP response payload. A verbatim URL is one that is present in a downloaded web object, and the app subsequently fetches exactly that URL. A modified URL is one that is present in a downloaded web object, but the app modifies the URL before fetching it. A partial URL is one where a simple string or portion of a URL is present in downloaded content, that the app uses to construct a full URL that it subsequently downloads.

type	URL
static	http://mobilefeeds.wsj.com/xml/rss/3_5557.xml
static	https://iwap.wsj.com/iphone/rolesByUuid?uuid=MSOFT_sagarwal&udid=WSJ-IPAD
static	http://mobilefeeds.wsj.com/xml/feed/v2/3_5549_2.rss
static	http://www.marketwatch.com/mw2/mediarss/wsjdn/wsjtv.asp?type=wsj-section&query=PersonalFinance&count=30
partial	http://online.wsj.com/xml/djml/http://blogs.wsj.com/five-things/2014/03/07⇓ /5-takeaways-from-the-february-employment-report/.xml
modified	http://mobilefeeds.wsj.com/xml/djml/SB10001424052702304732804579423454141533882.xml
verbatim	http://s.wsj.net/public/resources/images/BN-BV058_russia_G_20140307121641.jpg

Table 2.1: Some of the URLs fetched by a single run of the Wall Street Journal app.

```
[frame=single]
    ...
    <item>
    <title>5 Takeaways From the February Jobs Report</title>
    <link>http://blogs.wsj.com/five-things/2014/03/07/5-takeaways-from-the-february-employment-report/</link>
      ...
    <media:content xmlns:media="http://search.yahoo.com/mrss"
    url="http://s.wsj.net/public/resources/images/BN-BV058_russia_G_20140307121641.jpg"
    type="image/jpeg" medium="image" height="369" width="553">
      ...
    </item>
    <item>
    <title>More Uninsured Buy Health Coverage</title>
    <link>http://online.wsj.com/article/SB10001424052702304732804579423454141533882.html</link>
      ...
```

Figure 2.1: Snippets of the response payload from fetching the third URL in the Wall Street Journal example. The response was a 190 KB RSS file in XML format.

As a concrete example, we present the behavior of the Wall Street Journal app by Dow Jones & Company Inc. on a Windows 8.1 tablet. Table 2.1 shows some of the URLs that a single invocation of the app fetches. The first four URLs listed are static URLs, and are fetched everytime the app is launched on this tablet. The second URL contains a unique user identifier and a unique app identifier. Some of the contents of the third URL download are shown in Figure 2.1. Notice that the last URL in Table 2.1, which we classify as "verbatim" is embedded exactly as fetched in the middle of Figure 2.1. The top of the figure shows a URL for a news article, which the app modifies by prepending `http://online.wsj.com/xml/djml` and appending `.xml` – we call this a partial URL. The bottom of the figure contains a URL that the app modifies by replacing the hostname and the file type from .html to .xml – we call this a modified URL.

As the example shows, custom app logic that the app developer has built determines what URLs to fetch. That app logic may select a subset of URLs to fetch, and may construct or modify URLs from content that is downloaded from the web. This behavior is different from that of webpages in a browser, where HTML and other documents clearly demarcate URLs and common logic is followed by the browser to fetch those URLs. One exception is javascript code that executes in the browser, where potentially arbitrary logic could determine additional URLs to fetch.

2.2.2 Design goals

To guide our design of Airplane Mode, we identified the following goals for our system.

Download all web content that is relevant to the app. When a user is offline, she may behave differently in the app than otherwise. For example, when she is trapped in a long airplane flight, she may more extensively explore an app, than when she has a few minutes to kill while waiting for her sandwich at Subway. We want our system to strike a reasonable balance between fetching only that content that the user will need, and the performance and network overhead of fetching every possible URL that the app might request. In doing so, the system will need to handle all four types of URLs that we have identified.

Avoid app-specific logic. The system may incorporate a limited set of manually crafted heuristics, as long as those heuristics do not need to be modified each time a new app is released on the marketplace. Any app-specific behavior that is necessary for the system to operate should be automatically learned over time without human intervention. We attempt this goal in order to not make our system brittle with respect to which apps it supports. However, in the extreme, it may not be possible to support all apps – an app may generate URLs using pseudo-random number generators or cryptographic algorithms whose behavior we cannot predict.

Support apps that use authentication. A number of apps that we want to support may use authentication to download content or even specify in the URL what content to download. The New York Times app may only allow subscribers to access content, and the Facebook app relies on the user identity and authentication to select what content to download. Some apps may use cookies to authenticate legitimate requests. Our system should handle common types of web authentication, and maintain isolation between apps so that an app may not access another app's private web data through our system. However, it may not be possible to handle authentication where an app has a secret hidden in the app binary that is used as a key to sign outbound requests, such as in some uses of OAuth.

Limit the damage to battery, processing, and network performance. Ideally, our system should impose no more overhead than fetching URLs, parsing downloaded content to identify additional URLs to fetch, and then fetching those. The lower the overhead of our system, the more often a user or other automatic mechanism may trigger our system to fetch content for offline use.

2.2.3 Design choices

With these goals in mind, we have five candidates for the system design, of which we pick the last one below.

Rely on app developers to extend their apps to include such functionality. This design choice is the most flexible in potentially supporting any kind of app design. The underlying OS will need to provide the app developer with some interface through which she can specify what to fetch and the OS will execute that on behalf of the app at opportune times. The Windows 8.1 SDK includes such an API called *ContentPrefetcher*. This API allows the app developer to specify upto 40 URLs that the OS will fetch, or a link to a specially formatted XML file on the web that has these URLs. Unfortunately, this model does not fit the fetch–parse–fetch–parse model that apps have. Such a system could work for static URLs, perhaps verbatim URLs, but not modified URLs nor partial URLs. Other APIs, such as iOS *Power Nap* and *Newsstand* have worse restrictions. A related API that exists on most mobile platforms is push notifications. This allows a web service to alert an app on a mobile device that something happened (such as a post on my Facebook wall). These APIs allow the web service to send a very limited set of bytes (256 B on iOS, 1 KB on Android, 512 B to 80 KB on WP) to show the user a notification at the top of the screen. The user then has to click on the notification to launch the app, which will then download the full content related to the notification.

Scrape app binaries for URL strings. As we have identified, there are some URLs that the app fetches on every launch. Sometimes, these URLs are hardcoded as static strings in the app code, and other times it may be deterministically generated by app

logic. A design that scrapes the app binary for text strings that look like URLs may not always work, and at best will only identify static URLs, and not those URLs that are present in downloaded content.

Periodically run apps in the background. This design has two flaws. Merely launching the app may not trigger all the downloads that will be needed – the user may need to click on certain items, such as sports headlines, to download additional content. It may be feasible to learn a model of where in the app UI the user clicks on, and then replay those clicks. Even in that design, we argue that the performance overhead of launching and running the entire app is too high.

Periodically run apps in the cloud. This design mitigates the performance overhead of running the app on the phone or tablet in the background. Upon successfully running an app in the cloud, the URLs that the cloud invocation downloaded can be transferred to the mobile device. However, in addition to the logic that is embedded in the app binary, user customizations such as passwords or cookies or preferences have an impact on what URLs the app downloads. Without knowing what portions of the app's local storage corresponds to these settings, such a system would need to sync the app's entire local storage with the cloud, which could be a large overhead for some apps.

Learn what URLs apps download and periodically fetch those. This design requires the system to passively observe what URLs apps request, and the content of the reponses from the web server. The system may extract strings from this content and use machine learning to build a model to construct URLs that the app needs. This model can then be executed to fetch fresh data for an app. This design has the potential of limiting performance overhead to only that which is necessary. However, the extraction of URLs from content and learning how to construct subsequent URLs can be challenging, and needs to be complete and generically applicable across a variety of apps.

2.3 System design & implementation

We now present the design of Airplane Mode, where we follow this last design choice candidate. Our design passively observes what web transfers apps normally engage in. Later, when the mobile device is not in active use and is charging, it will build a machine learning model for every app's network behavior. Finally, if the user clicks on a button on the display requesting to go to into "airplane mode", our system will first execute the model for every app which will download all relevant content for each app, and then will turn off the wireless interfaces. If an app is launched by

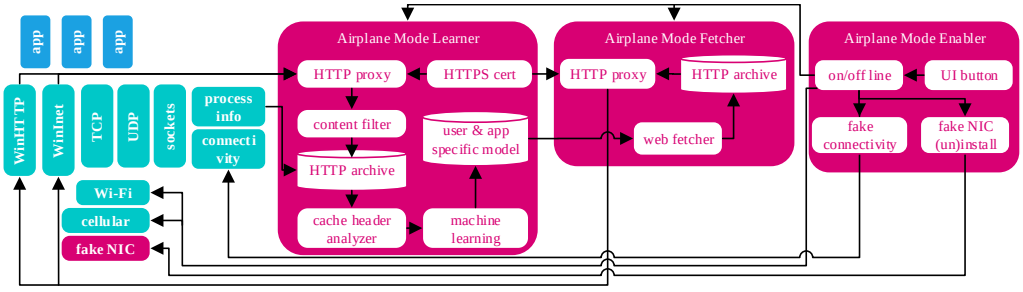


Figure 2.2: Airplane Mode system overview. All components run locally on the user's mobile device.

the user, we use a fake network interface to pass up previously cached responses to requests that the app makes.

We make the following assumptions. We target apps designed for information consumption, such as news, education, weather, and social networking. We do not target apps or functionality that commit transactions to web services, such as multi-player games and banking. We only consider network transfers that are HTTP GET and HTTPS GET – we do not model nor perform UDP transfers, HTTP POST, or other non-HTTP TCP transfers. We assume that HTTP GETs that apps perform do not alter state or semantics at the web service – this may not hold true if the web service owner parses HTTP logs for usage analytics.

Figure 2.2 shows an overview of Airplane Mode's architecture. There are three main components to our system: (1) the *learner* that watches app behavior and builds a model of what web content an app needs; (2) the *fetcher* that will execute the model

to download content that an app needs; and (3) the *enabler* that coordinates the two other components and fakes Internet connectivity for apps while disconnected.

2.3.1 Airplane Mode Learner

The learner passively observes the HTTP requests that apps make and the responses received from the remote web services. It does so using an HTTP proxy that the OS is configured to insert between apps and the underlying network stack. This HTTP proxy is enabled only when the OS has Internet connectivity. The requests and their responses are stored in compressed archive files in local storage on the mobile device. Both the headers and body of the requests and responses are stored, including any cookies and tokens that were used by the app in the outgoing request. Each request and response is associated with the identity of the app that initiated it. The HTTP archive is flushed periodically when it grows beyond a configurable size limit.

Periodically, when the mobile device is connected to power and is idle, this archive of prior web requests is handed over to a machine learning algorithm. We first separate the archive by app and invoke the machine learning algorithm separately on the data for each app. A separate ML model is created for each app. These models are used to identify which URLs to fetch to enable offline viewing. They are provided to the Airplane Mode Fetcher which will fetch content for apps at op-

portune times. The machine learning algorithm will learn from past app behavior which static URLs the app needs, and how to identify the verbatim, partial and modified URLs that the app needs. The input to this algorithm is the list of URLs that the app requested in each execution, and the response body from the web service. Sometimes, the requests are answered by the underlying OS's web cache, in which case we determine that situation by analyzing the cache options in the request header and using the appropriate body contents from the prior archived reponse. It is also possible that the app may explicitly cache content inside its isolated storage, outside of the OS's web cache. We handle this situation by identifying prior requests whose response cache header indicates they have not expired, and consider the situation that these URLs are still needed but cached explicitly by the app. This analysis is needed by our machine learning component, which we describe in § 2.4, where we formulate this problem as *multi-label classification* learning. Any cookies and tokens used in specific URL fetches are included in the machine learning model so that they can be used to fetch new content.

2.3.2 Airplane Mode Fetcher

When instructed to, the fetcher will execute the learned model for each app and download relevant web content and store it in an archive. That archive is indexed by app, and is provided to an HTTP proxy. When the OS is disconnected from the

Internet, this proxy will intercept outgoing requests by apps, and will respond with appropriate data from the archive.

The machine learned model for an app identifies which static URLs need to be fetched for the app. The web fetcher downloads those URLs. If the response body for any of those URLs is a text format such as XHTML, JSON, and XML, the fetcher will use a set of regular expressions to extract partial and full URLs and certain text keywords. This extraction is described in more detail in § 2.4. The machine learned model will also identify how to transform any of these partial URLs, full URLs, and text strings to URLs that the app needs. This may include appending or prepending text to the URLs or deleting portions of the URLs. We describe how these transformations are learned automatically in § 2.4.

These HTTP requests and their responses, including the header, body, cookies, and authentication tokens, are stored in an HTTP archive. When the mobile device is offline, an HTTP proxy will intercept all web requests that apps make. If the request is an HTTP(S) GET, the proxy will look it up in the HTTP archive to find a matching request with the same URL, cookies, authentication token, and user agent that the web fetcher made for the same app. If found, it will hand up to the app the response header and body from the HTTP archive. The app is unaware that such responses are actually coming from the proxy. If not found, or if the request was an HTTP

POST or other type of HTTP transaction, the proxy will return an HTTP 503 status code – service unavailable.

2.3.3 Airplane Mode Enabler

The enabler coordinates the different components of Airplane Mode. We use a simple UI button that the user presses to enable "airplane mode". First, the HTTP proxy in the learner is disabled. Next, the web fetcher is executed for every app that has a machine learned model. Once complete, the fetcher HTTP proxy is enabled. The wireless radios are turned off, and a fake network interface is installed. The OS is tricked into believing there is Internet connectivity, so that if an app checks for connectivity on launch, it will believe it is connected to the Internet.

When the user disables "airplane mode", normal operation is enabled. The Internet connectivity settings are restored, the fake network interface is uninstalled, the fetcher HTTP proxy is disabled, the learner HTTP proxy is enabled, and the wireless interfaces are turned on.

2.3.4 Implementation

We have implemented Airplane Mode on Windows 8.1 tablets. We have designed our system to not require any OS changes, nor app changes. It is packaged as

a Windows service and requires a one-time install by the user on their tablet using administrator privileges. Airplane Mode is implemented in C#.

For the two HTTP proxies, we use the FiddlerCore .NET class library [2]. FiddlerCore is the underlying engine to the popular Fiddler web debugging proxy. It is designed to be lightweight and can be incorporated into systems by calling appropriate APIs exposed by the library. The Windows 8.1 OS has two system libraries for requesting web content – WinInet and WinHTTP. We configure the OS to proxy all requests through either library to go through our proxies which are running on a local TCP port.

To minimize the storage overhead of the *learner*, we use a content filter to limit what web responses are stored in the HTTP archive. We eliminate the response body for any MIME type that is not text nor application. In this way, we eliminate images and media content that we would not parse as input to the machine learning algorithm.

To identify which app makes each request, we use a process monitor to identify which apps correspond to each active process ID, and query the HTTP socket properties for each outbound request to identify the associated process ID that created it.

[2] `http://www.telerik.com/fiddler`

To handle HTTPS requests by the app, our system generates a unique HTTPS certificate at install time and inserts it into the trusted certificate store in the OS. In this way, the FiddlerCore based proxy can inspect the contents of HTTPS reponses. We stress that these contents are used only to feed into the machine learning algorithm to identify what URLs to fetch, and neither these contents nor the models ever leave the mobile device. Our Airplane Mode service runs as a priviledged OS service, and hence non-administrator users do not have access to our system's process, memory, or storage.

The HTTP proxy in the fetcher uses FiddlerCore's *replay* mechanism to respond to outgoing app requests. Our code logic identifies which response body and header to use for each request, and the FiddlerCore library appropriately formats the response and passes it back up to the app

Windows 8.1 has a component called NCSI (Network Connectivity Status Indicator) which determines whether the OS is connected to the Internet via any of its enabled network interfaces. Apps can use an API call to determine whether Internet connectivity exists (typically at app launch) and this API call queries NCSI status. When the OS is disconnected, some apps will immediately throw an error at the user and provide no functionality until connectivity is restored. The fake connectivity component in the Airplane Mode Enabler reconfigures NCSI to effectively make it think that Internet connectivity exists when in fact it does not.

2.4 Machine learning formulation and design

The input to the machine learning component in Airplane Mode is a list of the HTTP GET requests that an app made in each execution session, and the responses from the remote server (or local OS web cache). From this input, it has to output a model for the app that can predict what static URLs the app needs, and how to generate the dynamic URLs the app needs from the content in HTTP GET responses. We break this problem down into a number of sub-problems: (1) identifying static URLs; (2) extracting relevant textual content from fetched web content; (3) correlating this textual content with actual URLs that the app fetches; (4) learning rules for transforming that textual content into URLs; (5) executing this model to predict what URLs an app will need to fetch.

2.4.1 Identifying static URLs

In § 2.2, we described *static* URLs as those that are hardcoded in the app's logic where the app will fetch these URLs deterministically on execution, or when a user clicks on an icon which triggers additional downloads. These are URLs that do not change, regardless of what content is downloaded by apps.

By observing HTTP GET requests from multiple executions of an app by the user, we can identify URLs that are repeatedly fetched by the app. This would in-

clude hypothetical URLs such as `http://www.cnn.com/topstories.xml` and `http://www.cnn.com/logo.png`, but not `http://www.cnn.com/Oscars2014-results.xml` which is an ephemeral URL that the app downloads only in some executions and then never again. Trivially, we may identify static URLs as those that are fetched in every execution of the app. However, there are two exceptions to this that have to do with caching. The "cache header analyzer" component of Airplane Mode, described in § 2.3, handles both the case where a HTTP request is answered by the OS out of the web cache inside the OS, as well as the possibility that the app has explicity cached the response inside its own isolated storage. In this latter case, where a particular URL is not fetched in a given run, it might be that a copy already exists in the app's storage, and the copy has not expired yet. This means that although the URL was not fetched, its content might still have been used by the app. We monitor the cache-control TTL directives of HTTP responses and keep track of each item's cache expiration time. If a previously fetched web object has not expired, we assume the app continues to use it and we include it in the input to the analysis of whether it is a static URL. If after it has expired, the app does not fetch it again, then we assume it is not a static URL.

There are two key factors that influence the success of identifying static URLs in this way. The number of runs of the app for which there is data to train on, and how far apart those runs are. If we have training data from only two executions of the app

by the user, where the executions are 5 minutes apart, then most URLs will appear to be static. That is because some emphemeral URLs will not have gone away that quickly (the third example above), and they will not have expired from the cache. In contrast, several executions that are hours apart should be sufficient to distinguish static URLs from emphemeral URLs.

2.4.2 Extracting embedded text

After identifying those URLs that an app has hardcoded in its logic, the remaining URLs are those that come from the web content that the app downloads. We described these embedded URLs in § 2.2 as *verbatim*, *modified*, or *partial*. As shown in Table 2.1 and Figure 2.1, some of these URLs are plainly present in downloaded content, some are modified from what is present in downloaded content, and some are constructed from strings.

Some of the relevant content may be present in the fetched response to static URLs, and some may be present in the fetched response to subsequent modified URLs. We parse the downloaded content, searching for text strings that may be relevant in constructing URLs. We consider only those downloaded content that have MIME types of text and application as identified in the HTTP response header, regardless of whether they are compressed or not.

file types	html, xml, xhtml, rss, soap, atom, text, json, javascript, x-javascript
patterns	url=“.+?”, “http:.+?”, “https:.+?”,href=“.+?”, src=“.+?”, <link>.+?</link>, “.+?”

Table 2.2: List of file types used to extract URLs and relevant strings from, and the regular expressions we use to extract them. Uppercase versions of regular expressions omitted for conciseness.

Inside these text files, such as the example in Figure 2.1, we may find URLs that are “cleanly” specified, beginning with a protocol delimiter such as “https://”. We also find URLs that do not begin with a protocol delimiter but instead begin with a tag such as url=”www.cnn.com”. Table 2.2 lists the file types that we parse, and the regular expressions that we use to pull out relevant strings. Some URLs have their slashes escaped, and then encoded in Unicode UTF-8 – we have regular expressions that detect such encoding and then decode those URLs. The last pattern pulls out any string that is within quotes, and is used only when the success rate of the next step of matching embedded text to fetched URLs is low. In that case, our extraction automatically assumes that string literals are being used to construct URLs, and then repeats the extraction to include any strings within quotes.

2.4.3 Matching embedded text to fetched URLs

In the set of strings that we have extracted from downloaded content, some will have been used by the app to construct and fetch subsequent URLs, while the rest

fetched URL	http://g3.imwx.com/TileServer/imgs/radar/u1398474900000/023012301.png
relevant embedded URL	http://gima.weather.com/1398474900000
embedded URL with longest substring match	http://i.imwx.com/

Table 2.3: Example where longest substring matching will identify the wrong embedded URL.

will have been ignored by the app. We need to identify which of these strings are relevant, and of those, match each of them to the actual URL that the app downloads. This matched list will serve as input training data for machine learning.

A common approach to this problem is to use longest substring matching. For every item on the left, it will evaluate how close it is to every item on the right by finding the longest substring that is in common between the two items. Unfortunately, this approach does not consider the uniqueness of those substrings, and fails in some cases to correctly match the relevant URLs in our dataset. Table 2.3 shows one such example. The top row is an actual URL that The Weather Channel app fetched during an execution, and the bottom two rows are two URLs that we extracted automatically from content that the app downloaded prior to fetching this top URL. Longest substring matching picks the 3rd row, while in fact the second row is the item we need.

We instead use a data-driven approach that relies on the unique *tokens* of a URL. We split each URL into its component tokens using a simple set of delimiters: / . _ ? =. In this way, the top URL in the table is split into “http:”, “//”, “g3.”, “imwx.”

grammar rule	URL transformation
concatenate	$U \Rightarrow S_{beg} + U + S_{end}$ $U \Rightarrow U + S_{end}$ $U \Rightarrow S_{beg} + U$
remove	$U \Rightarrow U - S$
substitute	$U_1 + U_2 \Rightarrow U_1 + S_{end}$ $U_1 + U_2 \Rightarrow S_{beg} + U_2$

Table 2.4: Transformation rules for converting an input string or URL into a new URL to fetch. "U" represents a string or portion of a URL from the input, and "S" represents a string or portion of a URL from the output.

and so on. We then calculate how frequently each unique token occurs across all the URLs. Those tokens that occur less frequently, such as "u1398474900000" get a higher matching score over more common tokens such as "imwx.".

For each URL that the app fetches which is not a static URL, we find the embedded string or URL that has the highest matching score, which is the most number of most unique tokens that match. This results in a list identifies for each URL that the app fetched (that we did not identify as a static URL), which string or URL embedded in previously downloaded content closely matches it.

2.4.4 A grammar for transformations

This list is now training data for automatically learning how to transform each embedded string into a URL that the app will download. To formulate this as a machine learning problem, we first define a set of rules for transforming such strings into URLs. In the example in Figure 2.1, we showed how some URLs that are

present in downloaded files are transformed by the app into new URLs that it fetches. It can delete tokens of the URL, it can insert tokens into the URL, or replace tokens of the URL. In Table 2.4, we formalize these three operators that can be applied to any input string.

As an example, we consider the URL `http://online.wsj.com/article/SB10001424052702304732804579423454141533882.html` from Figure 2.1. This URL was extracted from a downloaded file using our regular expressions. We automatically match it to this URL that the app subsequently downloaded: `http://mobilefeeds.wsj.com/xml/djml/SB10001424052702304732804579423454141533882.xml`. We tokenize both the input and the output URLs. The minimal set of transformations that need to be applied to convert this input URL to the downloaded URL is subsitute the first token (the host name) with mobilefeeds.wsj.com, subsitute the third token with xml, insert djml at the fifth token, insiert "/" right after that, and substitute the sixth token with ".xml". This is the machine learning problem we need to solve.

2.4.5 Common strings

By observing client app URL fetches, we determine that URLs often consist of a set of literals that are used by the app server to identify the features or the characteristics of the content that is requested by the client app. Below, we describe

the common literals or strings that are used in URL construction and describe how they affect Airplane Mode's functionality or success in reconstructing such URLs.

- Web API keys, such as `api_key=bvwp4jq9e6d3539k6vr4dhbd` and `width=50`. The client app will insert the value/s for each key, such as the value `50` for key `width`, into the URL to be fetched. Each key either has a single constant value, or varies within a constant set of values. This behavior is caught by our ML algorithm, as the ML either identifies the constant value, or captures the multiplicity of possible values using multi-label classification.

- Device descriptors, such as `platform=tablet`. Such descriptors do not change for a given client device. Our URL analysis captures this behavior.

- Geo coordinates, such as `postal=93111` and `location=40.71,-74.02`. In the first example, the user manually entered the zip code; this value rarely changes, and therefore our URL analysis captures this behavior. The second case is when the geo location value is dynamic. For dynamic geo location values, we perform a geo-lookup using the platform API, and then scan for latitude and longitude numbers that match. We account for the number of integer and decimal values, considering whether the decimal values are rounded or chopped. When we identify dynamic geo-location strings in URLs, we tag those URLs and re-resolve them when we need to fetch them.

- Timestamp. We handle timestamp values similarly to dynamic geo coordinates, where we tag and then re-resolve these URLs when we need to fetch them.[3]

- Random number, such as `CacheBuster=20140327`. Random numbers, by definition, are almost impossible to reproduce, and therefore we do not handle them.

In Table 2.5, we list the number of URLs that contain each of the discussed common string types, computed for each app as the median value per-run. We later show in our evaluation that we are able to handle these URL string types, excluding those containing random numbers, which is Newser fetches in our set of apps.

2.4.6 Multi-label classification

Unlike in traditional machine learning problems, in our case the *labels* applied to the input are unknown prior to training. These labels depend on the tokens present in the training data. Hence we first extract these tokens from the training data and formulate the machine learning on the fly. We have identified multi-label classification as the most relevant machine learning technique. We refer the reader to prior work [67, 105] on multi-label classification algorithms. Based on our experience in

[3]We note that timestamps included in URLs need to abide by the specified W3C date and time formats `http://www.w3.org/TR/NOTE-datetime`.

Table 2.5: Profiling common strings used in the construction of app URL fetches on a *per-run* basis, using traffic from heavy user browsing of app content.

Apps	Median # Fetched URLs per-run					
	Total	**API**	**Desc.**	**Rnd.**	**Geo**	**T**
Allrecipes	30	7	0	0	0	0
CNN	42	0	0	0	0	0
Everyday Food	172	0	0	0	0	0
Facebook	419.5	1	0	0	0	0
Flixster	7	5	0	0	1	0
Geek Trivia	2	1	0	0	0	1
HowStuffWorks	59	30	0	0	0	0
Khan Academy	29	1	0	0	0	0
Manga Z	198	40	0	0	0	0
Newser	105	39	0	11	0	31
The Wall Street Journal	85	9	1	0	0	0
The Weather Channel	100	47	0	0	22	0
Twitter	227	59	1	0	0	0
USA Today	174	109	12	0	0	0
500px	110	36	0	0	0	0

*API: Web API Keys, Desc.: Unique Descriptors
Rnd.: Random Number, Geo: Geo Coordinates, T: Timestamp

looking at training data, we believe HOMER is the most suitable multi-label classification solution [34, 106]. We use an existing implementation of HOMER from the MULAN library [106] in the WEKA [38] machine learning framework.

HOMER adopts an approach from prior work which transforms a multi-label learning problem into one or more traditional and commonly used single-label classification problems. Single-label classification learns from a set of examples that are associated with a single label ℓ from a set of distinct labels L. The output is then transformed back into its multi-label representation. To transform the problem, HOMER combines entire label sets into single labels to form a single-label problem. This method is referred to as a *label power-set* or *label combination method*. In the single-label problem, the set of possible single labels represents all unique combi-

nations of label subsets from the original multi-label representation. In our analysis of applied URL transformations, we find that for a given set of features of the input instance, multiple transformations (i.e. labels) are consistently applied. HOMER is able to exploit these recurrent combinations of label sets for given input instances to improve the accuracy of predictions.

The app-specific model generated by Airplane Mode's machine learning will map each input embedded URL or string "x" to the transformed, fetched URL(s) "y", such that $x \rightarrow f(x)$ and $f(x) = y$. In Table 2.6, we present an example of such a transformation from an embedded URL to its corresponding fetched URL(s). It shows that a set of *generic transformations* have been applied to the embedded string to obtain each fetched URL. These transformations are generic as they can be applied to subsequent embedded strings or URLs from future downloads by the same app. However, to achieve high accuracy, a learned transformation should be applied to "similar" embedded URLs or strings across multiple executions of an app. To constrain what embedded URLs or strings a learned transformation is applied on, we use a set of features to describe each embedded URL and use this feature set to distinguish different classes of embedded URLs or strings. These features are listed in Table 2.7.

Multi-label classification will learn from known examples, where each example is associated with a set of labels ℓ, from a finite set of disjoint labels L [67, 105]. In

Embedded String	Fetched URL	Transformation {Action, Location in URL or String}
http%3A//www.gannett-cdn.com/-mm-/03dc7ac085a4ddeb0f8644ece522000b88507521/c%3D228-30-2790-1959%26r%3D2048x1536/local/-/media/USATODAY/GenericImages/2013/09/01/1378039324000-public-AP-2013-BUDWEISER-MADE-IN-AMERICA-FESTIVAL—DAY-1-58089132.jpg	http://i.usatoday.net/apps/image?imageUrl=http%3A//www.gannett-cdn.com/-mm-/03dc7ac085a4ddeb0f8644ece522000b88507521/c%3D228-30-2790-1959%26r%3D2048x1536/local/-/media/USATODAY/GenericImages/2013/09/01/1378039324000-public-AP-2013-BUDWEISER-MADE-IN-AMERICA-FESTIVAL—DAY-1-58089132.jpg&width=358&height=268&type=fitsOutsideCrop	{**Add**, *Beg.*, "http://i.usatoday.net/apps/image?imageUrl="}, {**Add**, *End*, "&width=358&height=268&type=fitsOutsideCrop"}
	http://i.usatoday.net/apps/image?imageUrl=http%3A//www.gannett-cdn.com/-mm-/03dc7ac085a4ddeb0f8644ece522000b88507521/c%3D228-30-2790-1959%26r%3D2048x1536/local/-/media/USATODAY/GenericImages/2013/09/01/1378039324000-public-AP-2013-BUDWEISER-MADE-IN-AMERICA-FESTIVAL—DAY-1-58089132.jpg&width=100&height=73&type=fitsOutsideCrop	{**Add**, *Beg.*, "http://i.usatoday.net/apps/image?imageUrl="}, {**Add**, *End*, "&width=100&height=73&type=fitsOutsideCrop"}

Table 2.6: Example transformations $x \rightarrow f(x)$ of an embedded string to the corresponding fetched URLs.

feature name	feature definition
length	# of tokens between non-alphanumeric characters { '/', '_', '.'}
extension	URL suffix, such as .html or .jpg
hostname	server name where this content is hosted

Table 2.7: Characterizing features of embedded URLs or strings.

our case, each label ℓ is a transformation that can be applied to a given embedded URL. For the example given in Table 2.6, there are three labels or transformations that can be applied to the same embedded string. Finally, on the basis of the training set of data containing examples whose label membership is known, the learned *classifier* (i.e. prediction model) maps new input data or embedded URLs to the set of labels to which the URLs belong. Since apps each exhibit a unique set of transformations, we apply multi-label learning on a per-app basis, to yield a set of app-specific classifiers or prediction models.

2.5 Evaluation

We have built a working prototype of Airplane Mode on Windows 8.1. In evaluating our system, there are a number of questions we want to answer:

- how many static and dynamic URLs do apps fetch?
- how many of the dynamic URLs is Airplane Mode able to predict?

- how sensitive is our URL prediction accuracy to past user behavior when collecting training data?
- if Airplane Mode cannot predict *all* URLs, what is the impact of missing URLs on the user experience?
- how many bytes are consumed in downloading content for offline use?
- how long does it take to execute the model for identifying what URLs to fetch for an app?

2.5.1 Methodology

We collect HTTP traces from app launches to build and evaluate components of Airplane Mode. We first use these traces in order to identify the labels or transformations applied to the input embedded strings or URLs; these traces are referred to as the *training data*. An important factor to evaluate in the training data is the number of URL transformation mappings captured by the dataset (i.e. comprehensiveness), and this can influence the reliability of predictions from the generated ML model. We therefore collect traces with varying browsing load and evaluate the impact of browsing load on the efficacy of the *training data* and the performance of Airplane Mode. We prepare datasets with three different browsing loads, as shown in Table 2.8. These datasets collect traffic from (1) launching an app and closing it,

Table 2.8: Profile of training data used for ML.

Training Data Set	Period between runs (hrs)	# Clicks	Period between clicks (s)
No Browsing	5	0	-
Some Browsing	5	15	60
Heavy Browsing	5	50	60

without browsing through the app, *No Browsing*, (2) browsing moderately through the app, *Some Browsing*, and (3) heavy user browsing, *Heavy Browsing*.[4]

Concerning other features of our datasets, we find that the number of independent app launches as well as the duration between independent instances of app launch more clearly influence the list of static URLs identified. In both cases, and depending on the rate at which the app replaces old content, Airplane Modecan incorrectly label dynamic URLs as static. This means that Airplane Mode will incorrectly prefetch that content. We believe this issue can be mitigated by applying a straightforward, real-time optimization that updates the static list based on URLs fetched in future app launch, by comparing the static list against future app fetches and pruning non-existent URLs.

In the datasets we use in our evaluation, we space app launch instances such that most content has expired, therefore ensuring a minimum degree of variability between URLs fetched in independent app launch instances. As we can see in Figure 2.3, the time needed for the content generated by apps to expire is approximately

[4]We use Microsoft's Research Automation Tool (MSRat) to simulate user interaction with Windows apps by identifying and randomly clicking on enabled objects in the given app window.

4 hours. In Table 2.9, we profile the type of URL behavior we see for 15 typical apps during a single run, using the dataset with *Heavy Browsing*. The dynamic content comprises 19% and up to 95% of the URLs fetched per run, and it is therefore critical to identify this content for app operability. [5] Even with *Some Browsing*, dynamic content occupies up to 86% of the URLs fetched per run and from 21% up to 98% from *Moderate Browsing*.

Note that in our evalution, we exclude URLs that pertain to advertising content, telemetry, and other marketing content (such as services that offer detailed statistics about an app's traffic). We do so to solely evaluate the performance of the app itself. In most apps, excluding ad and marketing content does not interrupt or affect app operability or the user experience.

2.5.2 Multi-Label Classification Model

We have proposed multi-label learning as a prediction approach to web content prefetching for mobile apps, and we have discussed the implementation details of this learning algorithm in Section 2.4. We now evaluate the components in the design of our multi-label learner, namely (1) the accuracy in preparing the *training data* to the ML algorithm, and (2) ML accuracy in predicting the correct transformations to apply to new embedded URL or string input instances.

[5] The number of unique static URLs is sometimes less than the number of static URLs fetched by the app as some static URLs are fetched multiple times in a single app launch.

Table 2.9: Insight into app web traffic by profiling the median app behavior *per-run* from *Heavy Browsing*.

Apps	# Runs	Median # Fetched URLs per run		# Unique Static URLs
		Static	Dynamic	
Allrecipes	7	1	29	1
CNN	7	21	21	21
Everyday Food	7	9	163	7
Facebook	6	46.5	374.5	48
Flixster	7	4	3	4
Geek Trivia	7	1.5	2	1
HowStuffWorks	7	43	22	39
Khan Academy	7	7	15	7
Manga Z	7	75	123	75
Newser	7	39	101	6
The Wall Street Journal	7	23	63	21
The Weather Channel	6	71	32.5	40
Twitter	7	14	215	5
USA Today	7	18	156	8
500px	7	8	102	6

URL Extraction and Matching

Constructing ML models for a given system requires a set of examples or known instances in order to train the ML model. Preparing the *training data* for our app-specific, web prefetching models requires effectively extracting the relevant strings from HTTP payload content, followed by accurately matching the fetched URLs to the corresponding embedded content. We now evaluate this accuracy in mapping the fetched URLs to the correct embedded strings or URLs. This accuracy ultimately affects the prediction accuracy of the final ML model for future, unknown input instances.

In Table 2.10, we depict the accuracy of our URL extraction and matching techniques. We use the *Heavy Browsing* dataset, and from this dataset, we compute

the number of the fetched URLs that were successfully matched with URLs extracted from previous HTTP payload content. Note that we exclude static URLs from the total number of fetched URLs, as those are not embedded in payload content. These values are representative of the behavior we see from different browsing load datasets. We also find that *Some Browsing* is able to cover example URL transformations as comprehensively as a dataset collected from *Heavy Browsing*.

We observe that our URL extraction and matching methods achieve high accuracy. This accuracy reflects the success of our system in preparing the appropriate data to feed to the ML model. Some apps however, namely Geek Trivia and Newser, expose a limitation in Airplane Mode, which is that we do not handle URLs with dynamic content. Namely, we do not handle URLs constructed with variable content, using timestamps.

Now that Airplane Mode has matched URLs, the differences or transformations between matching URLs are then extracted, and Airplane Mode thus identifies the labels of the its ML classification algorithm. Next, we evaluate the accuracy of Airplane Mode's ML predictions, where the ML model is generated using the training we have just discussed.

Table 2.10: Preparing the ML *training data* using the *Heavy Browsing* dataset: accuracy of URL extraction and matching. Stats are reported over multiple runs.

Apps	# Runs	# of Fetched URLs*		Median % matched URLs per run
		Total	Successfully matched to extracted URL	
Allrecipes	7	269	197	68.4
CNN	7	133	127	100
Everyday Food	7	1710	1697	98.5
Facebook	6	2455	2359	96
Flixster	7	58	44	33.33
Geek Trivia	7	17	0	0
HowStuffWorks	7	292	277	95.5
Khan Academy	7	231	221	93.3
Manga Z	7	887	882	99.2
Newser	7	954	644	72.2
The Wall Street Journal	7	548	544	99.5
The Weather Channel	6	203	138	70.2
Twitter	7	1611	1223	78.6
USA Today	7	922	873	95
500px	7	706	703	100

* We exclude static URLs from the total number of fetched URLs.

ML Predictions

We adopt a multi-label learning algorithm from literature called HOMER.[6] After feeding HOMER the training described earlier, we now use the app-specific ML prediction models generated by HOMER and evaluate their accuracy in predicting fetched URLs.

In Table 2.11, we apply our HOMER-based ML approach and two alternative approaches to parse and prefetch URL content to the same traffic dataset, and compare their accuracy. The first alternative is a *naïve*, *general* approach that fetches URLs exactly as they appear embedded in HTTP response payload content. The second alternative is a *manual* approach where we manually evaluate the traffic generated

[6]The base classifier is an SMO design.

by an app and construct observation-based, app-specific rules to generate a list of candidate URLs to fetch. Since the *manual* approach requires detailed analysis of app traffic, we restrict our analysis period to 4hrs per app, and we implement it for four apps only.

Our design of a *manual* approach mostly outperforms the *naïve, general* approach, however it is impractical for two obvious reasons. The first is that a manual approach is not scalable, as specific rules need to be created for each app, supported by our analysis of app HTTP traffic. Second, by observing HTTP traffic alone, the user is bound to miss rules, as is clear for Twitter where the *general* apprach performs better than the *manual* approach. We observe that popular social networking apps that generate content and update content at a high rate are hard to emulate; in Table 2.9, we observe a high dynamic content count from Twitter. All in all, Airplane Mode clearly outperforms the *manual* and *general* techniques, and our solution is both scalable and generalizable.

In Table 2.12, we analyze the prediction accuracy of Airplane Mode in greater detail. We evaluate the impact from using different training data on predicting URLs for app launches with different browsing behavior or loads. In particular, we are evaluating the impact of browsing load on the completeness of the training data, and its ability to accurately predict URLs for different user browsing behavior with the app. We note that in our computation of the number of URLs that Airplane Mode

extracts from HTTP content, we consider all identified URLs, though those URLs might not be valid URLs.

We can see that the general trend in Table 2.12 is that prediction accuracy improves as the ML training data is derived from traffic with heavier browsing load, however not by much. This can be explained by the number of additional observations on URL transformations that the training data might cover with heavier browsing of app content. Also, as the training data is derived from datasets with heavier browsing loads, the number of static URLs identified increases. This observation indicates that not all static URLs are fetched on app launch, but are fetched after browsing the app and clicking on particular links with static URL identifiers, such as the sports page section.

We now take a closer look and categorize apps based on their behavior. USA Today exhibits unique behavior, whereby we are able to achieve high accuracy in extracting and matching URLs, as shown in Table 2.10, but the performance of the ML algorithm somehow falls a little shorter. The reason for this is the wealth of transformations that can be applied to a URL with the same features, however different subsets of these transformations are applied to any given URL. This phenomenon impacts how accurately the ML, namely here HOMER, is able to predict the correct set of transformations to apply to a given URL.

Table 2.11: Evaluation of the accuracy of the proposed prefetching approaches for a single app run.

Apps	General	Manual	Airplane Mode
CNN	0.81	1.0	1.0
The Wall Street Journal	0.49	0.74	0.77
USA Today	0.49	0.85	0.5
Khan Academy	0.58	0.81	0.94

Other apps rely heavily on static URLs to populate their pages or cache content for long periods of time. These apps are namely HowStuffWorks, Khan Academy, Everyday Food, and Flixster. As discussed in Section 2.4.5 and shown in Table 2.5, a certain set of apps (namely, Geek Trivia, The Weather Channel, Flixster and Newser) contain dynamic entries, such as a timestamp or geolocation. Airplane Mode identifies the format of these entries and is able to handle them, however, within the context of extracting generic URL transformation rules, the URL extractor of Airplane Mode is unable to effectively extract such URLs for those apps, and therefore the ML alone does not handle these cases. As a result, the prediction values from the ML alone indicates poor performance, however, Airplane Mode is able to identify and handle all these URLs using a separate learning mechanism that looks for generic timestamp and geolocation literals in URLs and correctly fetch those URLs, as described in detail in Section 2.4.5.

Table 2.12: Analysis of ML predictions per app launch, reported as the median across multiple independent runs. We evaluate the impact of the browsing load of the training data on the prediction accuracy of test suites with different browsing load.

Apps	{ML Training Data, Test Suite}															
	{No browsing, Some browsing}				{Some browsing, No browsing}				{Some Browsing, Some Browsing}				{Heavy Browsing, Some Browsing}			
	# URLs	AP	M	S	# URLs	AP	M	S	# URLs	AP	M	S	# URLs	AP	M	S
Allrecipes	55.5	19	79	2	7	86	0	14	56	88	10	2	54.5	87	11	2
CNN	39	56	5	39	30	50	0	50	39	62	0	38	39	62	0	38
Everyday Food	37	61	36	3	3	33	33	33	32.5	86	11	3	32.5	86	11	3
Facebook	279	49	33	19	179	70	3	27	283.5	57	25	18	259.5	54	33	13
Flixster	9	33	22	44	5	0	20	80	9	33	22	44	9	33	22	44
Geek Trivia	1.5	0	100	0	0	0	0	0	1.5	0	100	0	1.5	0	100	0
HowStuffWorks	52	13	10	77	42	0	14	86	58	14	7	79	44	25	9	66
Khan Academy	15	0	73	27	13	38	8	54	24.5	60	4	36	24.5	60	4	36
MangaZ	67.5	0	48	52	35	0	0	100	111.5	13	4	83	104	19	9	72
Newser	78.5	54	36	10	16	56	25	19	94.5	59	20	21	66.5	16	55	29
The Wall Street Journal	58	29	59	12	24	79	0	21	64	58	17	25	69	54	16	30
The Weather Channel	70.5	18	33	49	51	2	37	61	78.5	3	43	54	78	18	29	53
Twitter	58.5	11	86	3	6	33	33	33	105	8	56	33	78	13	68	45
USA Today	65.5	41	38	21	48	29	38	33	69.5	37	37	26	69.5	37	37	26
500px	63	10	87	3	8	75	0	25	60	80	16	4	60	55	42	3

*AP: % URLs Accurately Predicted, M: % URLs Missed, S: % Static URLs

2.5.3 System Performance

The overall performance of Airplane Mode, and its impact on app operability during periods of disconnectivity is evaluated in this section. We specifically support using ML in the context of HTTP app traffic prefectching by analyzing the time taken to generate and execte an instance of the ML. We then present typical behavior of Airplane Mode's prefetching mechanism by profiling the time taken and the bytes consumed by Airplane Mode to prefetch content. Finally, we take a close look at app's performance with Airplane Mode during offline periods. We find that our results are promising. Experiments to evaluate Airplane Mode are performed on a Windows Surface Pro with a 4GB RAM and a 1.7GHz quad-core CPU, running Windows 8.1.

ML model generation and execution overhead: In Table 2.13, we evaluate the time taken to build the ML model for each app, as well as the the time taken to identify all predicted URLs from a set of extracted embedded URLs or strings, i.e. the time to execute the generated ML. We include results from using training data from datasets of varying browsing load. We observe that the time to build an ML from any training data is within a permissable range, with a maximum wait time of $4s$ using traffic from *Heavy Browsing*. The time to form a prediction per-URL using the generated ML model is negligible.

Table 2.13: System evaluation of time to generate and to execute the model to predict fetched URLs per app-run, using training data from various browsing loads.

Apps	Time (ms)					
	Median to build ML			Median per-URL to execute ML		
	NB	SB	HB	NB	SB	HB
CNN	9	-	8	-	0	0
Everyday Food	26	163	527	0	0	0
Facebook	297	1752	3705	0	0	0
Flixster	-	4	7	-	0	0
Geek Trivia	-	-	-	-	-	-
HowStuffWorks	8	51	131	0	0	0
Khan Academy	3	56	75	0	0	0
MangaZ	6	131	1609	0	0	0
The Wall Street Journal	5	72	211	0	0	0
The Weather Channel	39	90	39	0	0	0
USA Today	16	129	338	0	0	0
500px	7	206	300	0	0	0

*NB: No Browsing, SB: Some Browsing, HB: Heavy Browsing

The cases where we do not have a time value to build the ML can be explained according to two reasons. When Airplane Mode identifies only a single transformation to apply to embedded strings or URLs, it is a straightforward process to apply this single label or transformation to embedded URLs to obtain the predicted fetched URLs, with no need for ML processing. This reason explains the lack of time values for CNN, Flixster, and Geek Trivia. The other reason for not having a time value is when the majority of the training data consists of static fetched URLs, or URLs that the app will not fetched, as an unexpired copy is cached in memory. This also contributes to the lack of time values for Geek Trivia, which is an app that is populated primarily by static content, as can be deduced from Table 2.9.

System overhead: We evaluate the impact of Airplane Mode on system performance in terms of the number of bytes downloaded, as shown in Table 2.14.[7] Specifically, we want to evaluate the system overhead. Using the collected HTTP traffic from our datasets, we load the traffic that is fetched from a given app launch to the memory of our proxy server. Airplane Mode then prefetches the additional content that it predicted. After an app is launched during offline periods, Airplane Mode will replay that content that the app requests that Airplane Mode has predicted correctly and stored in its memory. Airplane Mode will return a 404 error when the content requested by an is not stored in memory or is incorrectly predicted.

In Table 2.14, we list the total bytes that are downloaded for each app by Airplane Mode. We compute the number of bytes downloaded by a user, particularly characterized as a *Heavy Browsing* user, the number of bytes that are correctly predicted by Airplane Mode, and finally the number of additional bytes that are downloaded by Airplane Mode for offline viewing. We compute the number of bytes as the number of *as-transferred* data bytes. When we refer to *as-transferred* data, it is the number of raw bytes that are transferred by the app, whether compressed or not.

Firstly, we find that the majority of app content is in fact not compressed over all apps. This feature can be fully exploited through a cloud-driven approach, that

[7]The time needed to download the predicted URLs can be estimated using the speed of different wireless connections.

collects content on behalf of the app and therefore can bundle and compress the data for more efficient data transfer.

Secondly, we find that Airplane Mode can benefit from an add-on system that limits the amount of data that is downloaded. This feature would be specifically useful to limit the amount of downloaded bytes for a certain set of apps such as Everyday Food, MangaZ, and Khan Academy, where Airplane Mode is able to extract the full content dataset for that app's operation at any time, through analysis of the HTTP traffic. This can done by placing a restriction on the number of levels of HTTP traffic that Airplane Mode can explore and fetch. For example, one approach would be to restrict parsing and prefetching to two-levels deep, where Airplane Mode would parse and fetch content embedded in static URLs, and terminate prefetching at the content embedded in those fetched URLs. Another approach would be to learn user preferences and URL access patterns based on the user's interaction with the UI interface. Finally, we find that this additional content fetched by Airplane Mode enables the user to functionally browse the app and view important content during offline periods.

We note that we do not handle the AllRecipes, Newser, and Twitter app for limitations that we describe as open issues in Section 2.6. For the other apps that Airplane Mode is able to support that are listed in Table 2.14, we have proided a

service that enables an offline experience for apps from the content that is parsed, computed, then prefetched correctly.

Table 2.14: System evaluation of Airplane Mode from fetching URLs from one test run over Wi-Fi.

Apps	As-transferred Download Size of URLs (MB)			
	Total Fetched	**Heavy Browsing User**	**Correctly Predicted**	**Extra Airplane Mode**
CNN	78.53	1.53	1.53	77
Everyday Food	2,885.43	5.43	5.21	2,880
Facebook	288.5	4.5	3.4	284
Flixster	260.13	1.13	1.07	259
Geek Trivia	0.14	0.14	0.14	0
HowStuffWorks	123.85	27.66	2.49	96.19
Khan Academy	10,432.86	431.41	400.00	10,001.45
MangaZ	1,021.59	28.78	27.80	992.81
The Wall Street Journal	160.02	102.91	12.2	57.11
The Weather Channel	1.24	1.24	1.24	0
USA Today	2,589.69	159.60	159.35	2,430.09
500px	58.5	6.79	4.41	51.71

2.5.4 Impact of missing URLs on user experience

We evaluate the impact of Airplane Mode on app operability during offline periods. To do so, we capture and compare screenshots of an app window between connection periods and offline periods when running Airplane Mode. These screenshots reflect the user's experience of the app, and can be used to gauge how well our system works. We post a video demonstration of app usage while connected versus while disconnected on `http://cs.ucsb.edu/~laradeek`.

Our study of the user experience demonstrates that Airplane Mode is able to prefetch the URLs that are critical to display content and enable interactivity with

the app. The missed URLs are found to be dynamic URLs that link to secondary content for each app. We believe that using a more realistic user study, and by collecting HTTP traffic from real user browsing, a lot of these missed URLs can be avoided. We argue that the training data sets from real users would exhibit predictable behavior, as related work and studies have revealed the predictability of user behavior [118]. In such cases, a lot more instances of a transformed URL would appear in the training data, thus increasing the accuracy of the ML approach. As discussed in Section 2.8, we leave a real user study to future work.

2.6 Open Issues

We now discuss the issues associated with the application of Airplane Mode to existing systems.

2.6.1 Posts

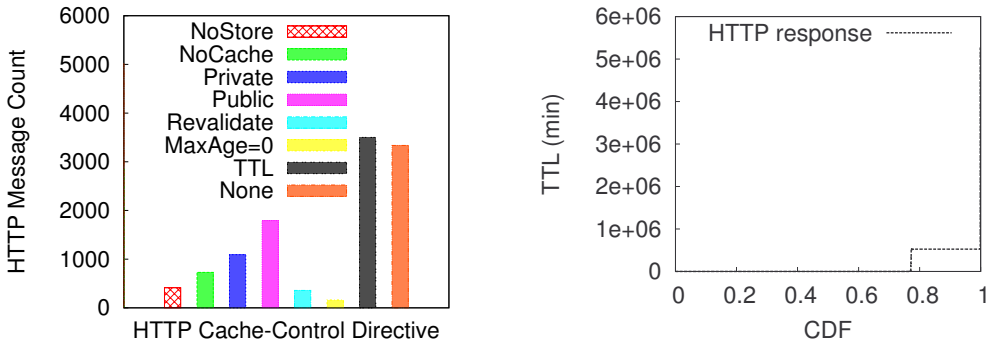
An inherent limitation of Airplane Mode is that it does not support any actions by apps that are interactive at a web service – for example, any HTTP POST requests by the app while disconnected will fail, which will prevent the user from posting to a Facebook wall. A key assumption we make is that any HTTP GET is idempotent, hence we assume we do no harm by issuing HTTP GETs when the app

is not running. This assumption may be wrong if the app developer has embedded POST semantics inside the GET or web server analytics change based on the frequency of HTTP GETs. We can potentially limit the negative impact of such an assumption by allowing Airplane Mode for only certain classes of apps – those in the "social", "news", and "weather" sections of the app marketplace, but not those in the "banking" and "gaming" sections.

Another limitation occurs when apps violate the HTTP standard specifications. Namely, the standard specifies that PUT should update a resource at a particular URI, and NOT return content `http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html`. However, apps such as Allrecipes embeds content in PUT statements, which is content that Airplane Mode skips based on the standard requirements. For Allrecipes that does not follow standardized rules, Airplane Mode would have to rely on a customized design in order to handle content stored in PUT statements, instead of GET statements.

2.6.2 Expired content

Between the time when Airplane Mode fetches web content for an app, and subsequently the device goes offline and the user launches that app, some of the web content may have expired. Figure 2.3 shows that a significant fraction of web content in the apps we consider do not have an expiration time set by the web service, and



(a) (b)

Figure 2.3: Left: distribution of HTTP cache-control directives in responses from web servers to a total of 7886 HTTP GET requests by apps. Right: CDF of TTL values in HTTP response headers where a TTL was specified.

of the ones that do, 20% of objects have a lifetime under 4 hours. If fresher content is not available, Airplane Mode will hand up expired content to the app. We believe in most situations this is preferable to not handing up any content to app, but this policy decision can be trivially reversed in our system.

2.6.3 Limitations in URL extraction and reconstruction

In Section 2.4.5, we identify the strings used to build URLs and point out that Airplane Mode cannot handle reconstructing URLs with random number entries. The impact of this limitation varies according to the relevancy of these URLs to app operability. As shown in Table 2.5, Newser uses random numbers in constructing certain URLs. We identify these URLs as Newser's "static" content, as identical

URLs with changing fields are consistently fetched on app launch. Since we are unable to reconstruct, and therefore fetch, the initial "static" content to populate the app, Airplane Mode does not support apps such as Newser. The impact of this limitation is alleviated if the respective content is subsidiary, rather than central, content to app operation.

Another limitation of our solution occurs when app URLs are embedded between non-standard and app-specific delimiters that our general URL extraction heuristics are unable to identify. For that embedded content, we are unable to identify the relevant URL to fetch.

2.6.4 Quantity of downloads

In designing the machine learning component in Airplane Mode, we do not pay particular attention to limiting what URLs are handed over to the fetcher for downloading. We believe that when users are offline, because they have limited access to online content (only what Airplane Mode has cached for them), they are prone to explore more of the app than they otherwise would while online. Hence we desire the system to download as much relevant content as possible, with little regard to what content within the app the user will find interesting.

To identify the apps to download content for, related work has shown that users browse a very limited set of apps on their mobile devices [80, 119]. Such a system

implementation can be used as an add-on or extension to Airplane Mode in order to prioritize which apps to prefetch traffic for, until the user is disconnected.

2.6.5 When to download

In this work, we do not solve the problem of deciding when to trigger fetching of web content. Our current implementation exposes a simple button to the user, which she clicks to cause Airplane Mode to start fetching content for offline use. We imagine that a context-aware system that has access to the user's calendar, location, and past network connectivity history, could learn when a device is likely to go offline and fetch content in advance of that happening.

2.6.6 OAuth

For apps that require authentication, some apps adopt the OAuth authentication strategy. Since Airplane Mode relies on extracting authentication information passively from HTTP content, it is unable to handle OAuth which embeds or hard-codes authentication tokens within the client-side app itself during installation. For apps such as Twitter that use OAuth, Airplane Mode would have to rely on a script that would actively ask the user for authentication information, rather than to handle authentication passively, in order to prefetch and serve the user content from those apps.

2.6.7 Cloud computation

An alternative approach to our system design would be to run the ML training in the cloud. The disadvantage of this alternative is that the user runs at the risk of sending and have their private data compromised in the cloud. Since ML training is not a time critical process in Airplane Mode, training can be performed locally on the machine when it is idle and charging.

On the other hand, performing the ML execution in the cloud may be more useful as this process is time critical. Similarly, the disadvantage of this approach is the danger of private information being gleaned from the generated ML model.

2.7 Prior work

The prior work most related to Airplane Mode is in browser caching. PocketWeb [66] is a caching system for speeding up web browsing on mobile phones. The user uses the PocketWeb app just like a regular web browser. PocketWeb may periodically predict that the user will visit a web page, and prefetches it within 2 minutes prior to the user actually visiting it. It does so by having the web browser visit that web page in the background. PocketWeb was inspired by PocketSearch [55] which caches popular web search queries on user's smartphones. In doing so, web search queries by users can be answered more quickly and perhaps without using

any cellular data. Armstrong *et al.* [6] solve the related problem of pushing web pages to mobile devices when those web pages have changed. They use a proxy system that can download a web page just like a web browser would and send it to the mobile client.

We address a significantly different systems problem when dealing with mobile apps. Unlike browsers, different apps can have different logic for determining what to fetch from the web. Apps are not attempting to interoperate with each other to provide the same experience to users like browsers are. Nonetheless, many apps use web servers for accessing content to display to the user. They often select which of that content to show to the user, and how to format it, using custom logic that is programmed into the app. A news app may fetch an XML file from the web that lists the latest news articles and some descriptive tags about each article. The custom app logic may select some of these to show to user. For those selected articles, it may decide to fetch a different representation of the article, by changing the suffix of the URL from .html to .json. In that JSON file, various images pertaining to that article may be referred to. The app may decide to use one of those images but pass it through a transcoding web service to resize the image.

FALCON [119] and PREPP [80] are two systems for speeding up app launch. They predict app launches before they occur and preload the apps into memory. The app then runs and may fetch network content. Our approach is fundamentally

different in that we do not launch apps. We instead learn what URLs apps may need and prefetch those without running any custom app code. We take this different approach for three reasons. We are not dependent on apps to prefetch all the network content they need on launch. We do not incur the performance and battery overhead of launching several apps in the background, and hence we achieve flexibility in how often we can download network content for an app. We can mimic connectivity and serve content to apps even if the mobile device is offline.

IMP [45] provides an explicit API for app developers to use in specifying what web objects to fetch. Underneath that API, IMP optimizes for application response time while dealing with battery lifetime and cellular data usage budgets. However, since apps typically fetch some web object, and then process it to determine what else to fetch, such a system cannot be used to provide a list of web objects to fetch without running the app code itself. IMP appears to be designed to optimize prefetching behavior *while* the app is running. Our goal is different in that we want to provide an offline experience for apps, and doing so requires fetching content even when the app is not running and hence we do not have access to the custom app logic for parsing content and constructing subsequent URLs to fetch.

CAMEO [50] is a middleware for predictively caching ads that may be displayed to the user if she launches a mobile app. CAMEO intercepts HTTP requests by apps for ads, and serves those out of its local cache. It observes past requests for ads by

apps to determine what future ads may be shown to the user. While similar in spirit to Airplane Mode, CAMEO does not solve the more general problem of arbitrary web traffic from apps. As such, it does not identify which app URLs should be considered, how they should be parsed from web content that is downloaded and how those should be transformed.

2.8 Conclusion

In order to *improve application bandwidth efficiency*, we designed Airplane Mode to provide a disconnected experience to users in some apps. By solving the extreme scenario of disconnectivity, we also solve the problem of network variability. In such cases, Airplane Mode allows us to prefetch and cache content over fast and free connections, and avoid expensive and slow connections.

In the design of Airplane Mode, we are inspired by the Coda [95] distributed file system for disconnected operation, but unlike a distributed file system where an entire directory may be made available offline, doing so for an entire website is not feasible. Given the unknown logic inside apps with regard to what web objects the app needs, we solve the challenge of automatically identifying the content that needs to be fetched in advance of disconnection.

Airplane Mode is the first system to solve the challenge of proactively caching web objects that apps need. Prior work has addressed the problem for browsers, which is a more tractable problem given the standard and predictable nature of web browser behavior in fetching content. Our system design addresses the problems of monitoring what content apps fetch during normal behavior, using machine learning to build a model of how to parse and transform text and URLs in downloaded content to the actual URLs that the apps need, running the model to fetch relevant content, faking Internet connectivity while offline, and handing up previously fetched content to the app as though it is coming from the origin web service.

In this chapter, we do not evaluate user satisfaction. How much value do users get from an offline experience in some apps (primarily news, social, and education apps)? Do users understand that certain actions (such as post to a Facebook wall) cannot be performed while offline, or do they find that failure jarring and detracting from the overall experience? Do users actually browse more content in the app while they are offline? We hope to perform such an HCI study on Airplane Mode in the near future.

Finally, we have presented Airplane Mode: a system that allows more efficient and controlled usage of bandwidth for widely and increasingly used mobile apps. Recent work has shown that despite raw WiFi speeds being very high, in reality users get very low effective throughput [48]. This is probably due to several reasons

including poor signal strength, over-crowding, and poor backhaul to the Internet. Improving WiFi itself, and time-shifting network transfers, as we do in Airplane Mode, together will reduce wireless contention and over-crowding and provide a uniform or consistent user experience, and allow for more efficient usage of network bandwidth.[8]

[8] In Chapters 5 to 7, we discuss our contributions towards improving WiFi itself through effective and efficient exploitation of next-generation, high-bandwidth technologies added to the WiFi chipset.

Chapter 3

Concurrent Use of Heterogeneous Networks

3.1 Introduction

The main thrust in wireless connectivity was established with the rapid growth of IEEE 802.11-based WLANs as the primary source of Internet connectivity for users. Large-scale commercialization of WiFi technology resulted in the availability of low-cost IEEE 802.11 radios and chipsets. WiFi soon became, and continues to be, integrated in almost all consumer electronic devices, not only in laptops, but also in cellphones, smartphones, and numerous other wireless gadgets and portable devices. This means that portable devices now commonly come equipped with multiple, high-bandwidth wireless network interfaces, particularly both a cellular and a WiFi interface.

Along with the widespread integration of multiple heterogeneous wireless interfaces onto portable devices is the increasing deployment of wireless network technologies around us. With the emergence of these two phenomen is the increasing opportunity to enable multifold increases in capacity in wireless physical links through the exploitation of the availability of multiple heterogenous network connections in parallel. Due to network heterogeneity, in terms of delay, bandwidth, and monetary cost, the major challenge in such situations is determining the way by which these networks can be utilized to better serve different network applications as well as different user requirements in terms of cost.

In this chapter, we address this challenge, and we propose and evaluate a system that exploits multiple heterogeneous networks in parallel in mobile wireless environments. More specifically, we propose a dynamic channel allocation mechanism that adapts to the state of the available channels to provide more efficient usage of network connectivity. We do so by observing channel throughput, creating a set of channel usage combinations, and then choosing the most efficient combination.

The *ParaNets* network architecture is an early effort to use multiple heterogeneous networks in unison [41]. The system we propose in this chapter builds on ParaNets to transparently unify available network connections into a single Internet service. Specifically in this chapter, our goal is to provide the best possible perfor-

mance subject to the cost of using individual networks and in response to varying network conditions.

Heterogeneous parallel networks, such as cellular, satellite, and Wi-Fi, are characterized by different bandwidth, delay, and monetary cost. These characteristics make each of these networks best-suited for specific types of data transmissions. For example, it may not be frugal to use high-cost satellite bandwidth for a movie download when a free Wi-Fi connection is available. Similarly, a user may not want to search for a free access point to send a short email when also already connected to a cellular data network. When multiple heterogeneous networks such as these are available, the major challenge is to determine the efficient use of each channel with respect to user performance expectations. The system we propose in this chapter dynamically determines efficient usage of available parallel networks based on traffic requirements as well as the state of these networks to provide a unified Internet service.

We build our system over the ParaNets-Enabled Data Bundling System for Intermittent Connectivity (DBS-IC) architecture, originally developed to improve communication over challenged networks [41]. ParaNets-Enabled DBS-IC takes advantage of the availability of multiple heterogeneous networks in parallel to create a perception of constant connectivity in challenged networks. ParaNets-Enabled DBS-IC statically distributes data over multiple channels based on the expected per-

formance of the channels' underlying technologies, as well as traffic bandwidth and cost requirements. While the static data allocation schemes achieve improvements in transmission delay and per bit cost, they suffer when underlying channel performance fluctuates. Our goal in this chapter, therefore, is to address the inflexibility of static channel allocation strategies.

The approach presented in this chapter is a dynamic channel scheduling mechanism suitable for use as the ParaNets-Enabled DBS-IC allocation strategy. Instead of a static allocation strategy as originally proposed, we present a technique that transmits varying amounts of data over all available channels based on actively measured channel conditions. The amount of data allocated to particular channels is referred to as a *data-to-channel* allocation.

We use the effective throughput of each of the parallel network connections as a determining factor to identify the best allocation strategy. Throughput is the average net bit rate delivered to the application layer and includes the effects of lower-layer factors such as packet loss, transmission delay, and channel congestion. In response to the throughput performance offered on each network, our solution dynamically adjusts the ParaNets-Enabled DBS-IC data-to-channel allocation strategy to achieve the highest possible goodput within the user-specified data transmission cost.

We emulate an implementation of our system on Emulab [110]. We configure different channels within the Emulab framework to act as satellite, cellular, and Wi-

Fi networks according to expected throughput, delay, and intermittent connectivity characteristics. We evaluate the effectiveness of dynamic data-to-channel allocation strategies in response to varying channel available bandwidth and availability specific to each emulated network technology. We also compare the performance of our dynamic data-to-channel allocation strategy with that of its static predecessor under varying network conditions and cost requirements. Our results show that the dynamic scheme outperforms the static scheme in all cases. At low costs and extreme channel conditions, the difference between the static and dynamic approach is small; however, as cost increases and under less extreme channel conditions, the dynamic approach exhibits a notable performance improvement.

The remainder of this chapter is organized as follows. In Section 3.2, we discuss the related work. Section 3.3 presents our system architecture and its operation. In Section 3.4, we describe our evaluation environment. In Section 3.5, we present our results. In Section 3.6, we discuss our future work. The chapter is concluded in Section 3.7.

3.2 Related Work

We begin this section with work that has addressed challenged network environments and systems that have been designed to deal with such challenged envi-

ronments. We then refer to research that has dealt with communication over heterogeneous networks, namely satellite, cellular, and Wi-Fi networks. Finally, we discuss work that has addressed allocation strategies to improve performance over communication channels.

A challenged network was first defined by Kevin Fall as a network that experiences mobility, frequent disruption of network availability, high round trip time, and high packet drop rate [24]. Disruption Tolerant Networks (DTNs) are a form of challenged networks. DTNs deal with hostile environments that are characterized by the lack of a continuous connection, and the inability to establish an end-to-end routing path thereby causing long delays[1]. Researchers have studied DTNs with a major focus on routing issues in such extreme environments [64]. While our system focuses on delivering bundled data from a wired stationary device to a mobile node, the same bundling and opportunistic delivery concepts are applicable to "pure" DTNs.

A recent focus of several DTN projects has been to address and study transport layer issues in extreme networking environments as well as focusing on forwarding techniques and routing algorithms. Our work takes a different approach from DTN's by taking advantage of any available network and developing a dynamic approach to efficiently and intelligently utilize alternative network paths in the data transmission

[1] http://www.dtnrg.com/

process. Instead of considering only the target DTN network, we consider all available networks and use a combination of available network connections in parallel.

ParaNets is a recently proposed architecture which utilizes end-to-end networks in parallel with an intermittent wireless connection to best serve a challenged network [41]. By taking advantage of other networks, such as cellular and satellite networks, in addition to an opportunistic Wi-Fi connection, connectivity and communication can be made more reliable and efficient. In ParaNets, the cellular and satellite networks are treated as channels that can be used depending on the size and type of the message. These "alternate" networks are particularly well suited for control information. ParaNets simulation results on top of DTN show that the use of alternative channels of communication when the Wi-Fi connection is broken significantly improves the performance of the challenged network, and hence, user-perceived performance [41].

The ParaNets-Enabled DBS-IC system on which we build our dynamic channel scheduling strategy is an architecture that takes advantage of intermittent connection opportunities as well as infrastructure networks in parallel [23]. ParaNets-Enabled DBS-IC adopts DBS-IC as the challenged network architecture. DBS-IC predicts which data the user will request and prefetches the data before the user explicitly requests it, thereby attempting to create an image of constant connectivity [46]. Additionally, DBS-IC proactively takes advantage of periods of connectivity by group-

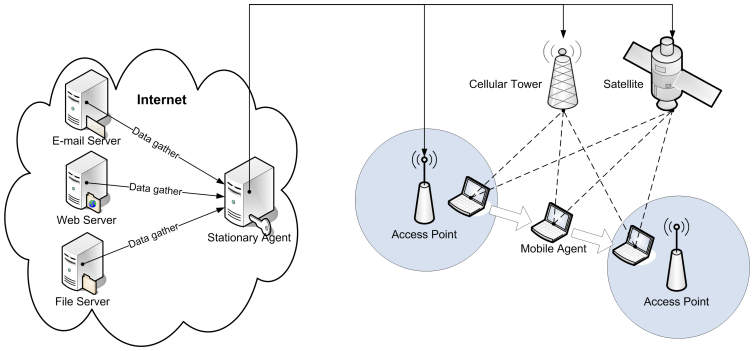


Figure 3.1: The architecture of a ParaNets-Enabled DBS-IC system.

ing user-requested data into a packet called a *bundle* and delivering it to the user incrementally in the form of prioritized *mini-bundles*. ParaNets-Enabled DBS-IC allows DBS-IC communication over multiple heterogeneous networks in parallel. However, it does not address methods to efficiently deliver data over ParaNets–the focus of this chapter.

Research on the integration of heterogeneous networks - namely cellular, satellite, and Wi-Fi networks - has focused mostly on inter-system handoffs and on handling inherent TCP/IP unsuitability to challenged environments [11, 52]. CAP and APHOHN are examples of architectures built for such purposes [54, 61]. In these environments, however, the networks are not utilized to best serve the intermittently-connected network. Other research has discussed channel allocation schemes and methods to improve overall network performance; yet, this work has largely concentrated on dealing with single channels [63, 121]. In our work, we devise appropriate allocation strategies between multiple channels at a time; furthermore, we

incorporate application layer parameters such as user preferences, size and type of data, or cost to provide realistic and suitable recommendations for dynamic channel scheduling.

This work distinguishes itself from previous work in that it examines dynamic methods to efficiently exploit parallel networks. The dynamic portion of our system is based on parameters such as the conditions and state of the underlying network channels as well as user-defined transmission cost constraints. We identify and quantify the benefits of ParaNets and, in particular, how effective and important its use is in challenged network environments.

3.3 System Architecture

In this section, we present an overview of our proposed approach. We first describe the system architecture of ParaNets-Enabled DBS-IC and its static data-to-channel allocation mechanisms. We then explain the operation of the dynamic channel scheduling algorithm proposed in this chapter.

3.3.1 ParaNets-Enabled DBS-IC

Our proposed system is built over ParaNets-Enabled DBS-IC depicted in Figure 3.1. The *stationary agent* maintains Internet connectivity and acts as a caching

proxy for a *mobile agent*. The role of the stationary agent is to prefetch and deliver the information requested by the mobile agent. The mobile agent is located on a mobile device in motion between access points. Since access points provide coverage in a limited area, mobile agent mobility results in intermittent Wi-Fi connectivity. In addition to Wi-Fi connections, the mobile agent may also be in range of satellite and cellular networks which provide wide area connectivity. The details and characteristics of the Wi-Fi, cellular, and satellite connections we emulate are detailed in Section 3.4.

The stationary agent prefetches data for each mobile agent based on (1) a predictive method whereby the stationary agent keeps a history of the information the user has requested to view in past sessions, and (2) a method whereby the mobile agent explicitly informs the stationary agent of the web, e-mail, or file data it would like to access. The stationary agent then contacts the corresponding application servers and gathers the requested information.

The stationary agent bundles the gathered data into a set of mini-bundles to be divided for transmission onto the available channels according to our proposed data-to-channel allocation strategy. An alternative to using mini-bundles is the aggregation of data into a single bundle. However, mini-bundles allow for prioritized transmission of content grouped into high priority mini-bundles and for straightforward reassembly of data received over different channels. Mini-bundle also simplify the

dynamic evaluation of candidate data-to-channel allocation strategies as described in Section 3.3.2.

The static data-to-channel allocation strategy is determined as follows. The static strategy calculates the throughput and cost values of all the possible data-to-channel combinations based on ideal network performance, where performance reflects network throughput under ideal channel conditions. Users specify the maximum cost in cents/MB that they are willing to incur to receive data: the *user-defined cost*. The static mechanism will then calculate all possible data-to-channel mappings using the user-defined cost as a threshold. The range of points that the static strategy will scan are those that incur the user specified maximum cost or a lower cost in close proximity to that maximum cost. Within this range of points, the static scheme will select the transmission strategy that achieves the highest throughput. By repeating this process for different user-defined costs, the static strategy will create a set of allocation combinations that it will use to build a data-to-channel mapping.

Figure 3.2 shows one possible channel-share-to-cost mapping based on individual channels' usage price and expected bandwidth where the x-axis specifies the user-defined cost. We note that the behavior of the static mapping strategy shown in Figure 3.2 varies. For example, the use of the cellular channel fluctuates whereas that of the satellite behaves like a step function. This behavior reflects the selective approach that the static method follows to determine the data-to-channel allocation

mapping. Methods for channel usage are based on finding and selecting the data-to-channel allocation combination that achieves the highest throughput within a particular cost range that is specified by the user-defined cost. In other words, there are no fundamental network conditions for channel usage. The static mechanism assumes that the respective channels will behave ideally in all situations, which is far from realistic.

Based on a simple survey of currently available service plans, we use a cellular data plan of 0.50 cents/MB at 14.4 Mbps[2], and a satellite plan of 0.65 cents/MB at 10 Mbps[3]. This cost model reflects the higher per bit cost of cellular and satellite delivery as well as their greater expected availability. Users willing to pay more for use of these networks generally experience lower download times as a larger share of their data is sent over more frequently available cellular and satellite networks. On the other hand, users willing to pay less, will take greater advantage of the opportunistic Wi-Fi connections, but may need to wait for such connections to become available. In such situations, any time critical mini-bundles are transmitted using cellular or satellite channels. Based on the user-defined cost for data transmissions, the stationary agent looks up the share of mini-bundles to be sent on each channel in the static mapping and transmits data accordingly.

[2]http://www.t-mobile.co.uk/
[3]http://www.wildblue.com/

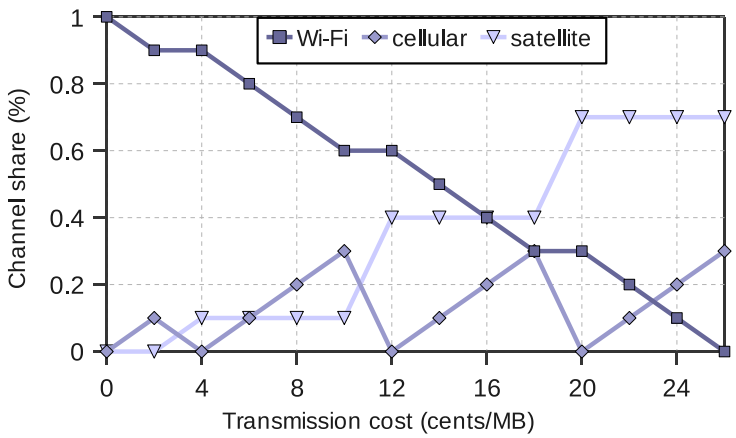


Figure 3.2: Static data-to-channel allocation mapping.

While the static mapping between the cost and channel share has been shown to have shorter download times at lower cost when a single connection is used, the solution suffers when the performance of the underlying network changes. For example, the throughput of the Wi-Fi connection can vary with respect to the load or availability of the access point. Similarly, cellular connection throughput can depend on signal strength [111]. To deal with changes in network conditions, a dynamic mechanism to determine channel data share is needed. Our extension to ParaNets-Enabled DBS-IC monitors available bandwidth on each channel and adjusts the channel-share-to-cost mapping dynamically and then uses that mapping to choose the beset data-to-channel allocation strategy.

3.3.2 Dynamic Data-to-Channel Allocation

The earlier version of ParaNets-Enabled DBS-IC adopted a static approach of utilizing the parallel networks. We now propose a mechanism to adjust the data-to-channel allocation strategy to changing network conditions using a dynamic channel scheduling algorithm.

The static data-to-channel allocation strategy of ParaNets-Enabled DBS-IC works well when network bandwidth adheres to advertised performance. However, in real network deployments the offered bandwidth will vary with changes in channel load, signal strength, and intermittent connectivity. Subsequently, the performance of any static allocation strategy will degrade when available bandwidth diverges from the amount of bandwidth assumed when the static calculation was performed. To assure efficient operation of ParaNets-Enabled DBS-IC, the data-to-channel allocation strategy needs to dynamically adjust in response to changes in network conditions.

Given that network access is assumed to have non-zero cost, the goal of our dynamic data-to-channel allocation strategy, therefore, is to find the allocation with the highest goodput within a given cost limit. To select the appropriate data-to-channel allocation mapping, we evaluate the goodput performance of candidate allocation mappings with respect to their cost. The set of candidate allocation mappings is the set of all possible ways of dividing the queued mini-bundles across the available

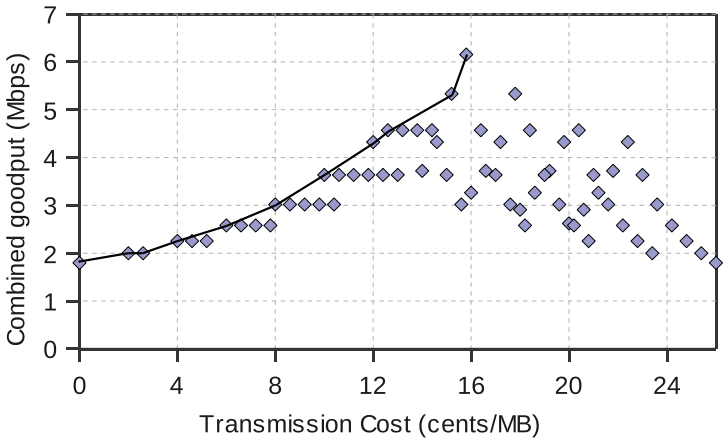


Figure 3.3: Combined channel goodput of data-to-channel allocation mappings.

channels. Thus, one particular allocation mapping for ten mini-bundles might be to send five mini-bundles on the Wi-Fi connection, three on the cellular connection, and two on the satellite connection. To evaluate the relative performance of candidate allocation mappings, we maintain a running average of channel goodput based on past transmissions. Channel throughput reflects the state and condition of the underlying network such as network load, packet delay, packet loss, or temporary loss of connectivity. By considering channel throughput, we are considering network factors relevant to bulk data transfer performance.

To choose a data-to-channel allocation mapping, we use the measured channel available bandwidth and calculate the expected combined goodput of each candidate mapping with respect to cost. The results of a sample calculation are presented in Figure 3.3. Data in Figure 3.3 exhibits a bell-shaped plot of goodput versus trans-

mission cost. At zero cost, only the the Wi-Fi channel is used. As the cost limit increases, the mapping gradually shifts transmission of data to the more expensive cellular and satellite networks. This shift in data-to-channel mappings increases network throughput as networks are exploited in parallel in order to speed the transmission of data. This trend, however, is inverted beyond a point where maximum goodput is achieved and whereby additional use of the alternative low-bandwidth channels will not increase throughput, but only increase the cost.

The data-to-channel allocation mapping is chosen as follows. Based on the distribution of goodput combined with cost, the dynamic mechanism chooses the best data-to-channel allocation mapping as the one with with the highest goodput. The dynamic mechanism recurses by finding the next best point located to the left of the previous value. The set of these points are then connected together. The resulting line, shown in Figure 3.3, forms the upper left boundary of the bell-shaped curve. It follows that all the data-to-channel allocation mappings below and to the right of this set of points represent a lower goodput-to-cost ratio and should not be chosen. Out of the set of selected points, the dynamic channel scheduling algorithm chooses the best points below the per megabyte cost a user is willing to incur.

Figure 3.4 shows the dynamic data-to-channel mappings for the set of selected points from Figure 3.3. For each of the selected points in Figure 3.3, Figure 3.4 shows the fraction of mini-bundles that are sent using each of the three available

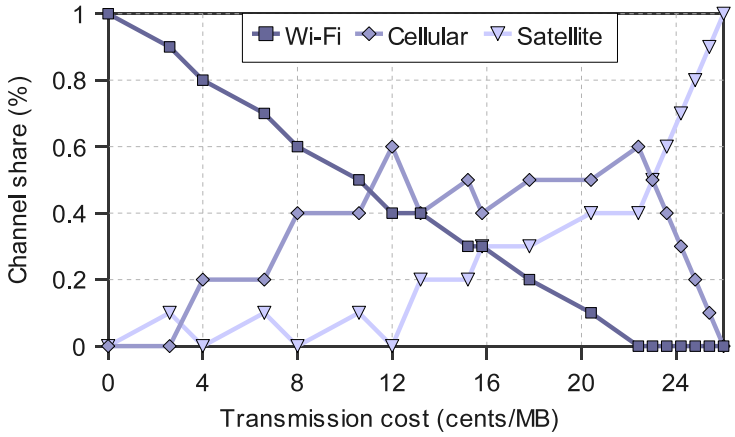


Figure 3.4: Dynamic data-to-channel allocation mapping.

network connections. For example, for the lowest transmission cost, all bundles are sent using the Wi-Fi connection; for the transmission cost of "12", 60% of the mini-bundles are sent using the cellular network, 40% are sent using the Wi-Fi connection, and none are sent using the satellite network. For completeness, we calculate and include in Figure 3.4 the set of points to the right of the highest goodput point.

The results of Figure 3.4 complement the findings of Figure 3.3: to increase combined throughput, parallel networks must be exploited for data delivery. A consequence of the increased throughput, however, is increased cost. As cost increases (moving along the x-axis), the data-to-channel mapping shifts from complete use of the Wi-Fi channel to more use of the cellular and satellite networks.

The set of points in the data-to-channel mappings presented in Figure 3.4 varies with varying network conditions. Changes in network conditions affect the location

of points on the graph. The result, therefore, is a change in the resulting data-to-channel mapping and a different set of selected points. For example, if the Wi-Fi channel conditions deteriorate, the goodput will decrease for combinations with Wi-Fi transmission, and the dynamic data-to-channel mapping will determine the efficient points that will express a decrease in Wi-Fi channel usage at each cost value.

The mapping in Figure 3.4 represents the allocation mappings selected even when channel bandwidth changes dynamically. The dynamic scheduling strategy used to produce this mapping differs compared to the static mapping in Figure 3.2 where the allocation strategies are based only on the advertised channel bandwidth values. The *static strategy* builds its data-to-channel allocation mapping based on ideal conditions for network performance. The static mechanism selects transmission combinations that represent the highest throughput within the range between the user defined cost and the costs in its lower proximity. The *dynamic strategy*, on the other hand, selects the efficient points based on continuous network measurement, as shown in Figure 3.3. Further, when the cost metric exceeds the optimum point at a cost value of 16, the static approach would continue to operate in the high-cost region whereas the dynamic approach will remain below this threshold. The dynamic approach will not operate beyond this threshold point because it would incur additional cost but with decreased throughput. This result is shown in Figure 3

by the corresponding drop in the second half of the bell-shaped curve. These data points indicate that at costs higher than the threshold point, the dynamic strategy always performs better.

We observe that the point distribution of the dynamic strategy in Figure 3.4 is more well-behaved in comparison to that of the static strategy in Figure 3.2. At low costs, both schemes behave similarly. However, as the allowable cost limits are increased, the dynamic approach follows a more continuous pattern that is representative of transitions between channel usage that are based on realistic network conditions. This continuity implies that a change in the use of a particular network will be based on knowledge of the actual network conditions. For example, if the satellite connection is weak, channel usage over the satellite network will be incremented only after a particular increase in the allowable cost and not earlier. The static mapping, however, is not aware of actual network state and its mappings are based on favorable, unloaded conditions; therefore, it does not always exhibit effective balancing across all of the underlying channels. Ultimately, with an increase in network channel conditions, the differences between both approaches becomes more apparent as the dynamic approach is more adaptive and offers efficient data-to-channel mappings. A detailed comparison between the static strategy and the dynamic strategy is presented in Section 3.5.

The advantage of our dynamic strategy is the data-to-channel allocation mappings that adjust to changing network conditions. Additionally, unlike in the static case, our dynamic strategy always chooses values left of the threshold point in the bell-shaped curve, whereas the static strategy might choose a high-priced solution, while a lower priced, higher combined goodput allocation strategy is likely available. When a user's service cost is fixed, the best strategy is the point with the highest aggregate goodput. However, when a user's service cost is usage-based, the user will likely be interested in choosing a lower performance and price point, thereby reducing use of more expensive channels by still achieving reasonable throughput and response time.

3.4 Evaluation Environment

In order to evaluate the performance of our system, we have created an emulation environment that enable us to evaluate the performance of the ParaNets-Enabled DBS-IC static and dynamic data-to-channel allocation strategies. The main configuration parameters for our evaluation are presented in Table 3.1. We choose the parameters that best represent the behavior of the underlying cellular (3G), satellite (LEO), and Wi-Fi (802.11g) communication channels. We evaluate our system un-

Table 3.1: Emulation configuration.

	Parameter	Value Range	Nominal Value
Application	Prefetched data	10 MB - 20 MB - 40 MB - 100 MB	40 MB
	Number of mini-bundles	10	10
Wi-Fi	Data rate	54 Mbps	54 Mbps
	Round trip time	(60 ± 10) ms	(60 ± 10) ms
	Ambient load	0% - 50% - 90%	50%
	Intermittent connectivity Model	15s on, 30s off; 20s on, 10s off; 40s on, 15s off; 120s on, 20s off	20s on, 10s off
Cellular	Data rate	14.4 Mbps	14.4 Mbps
	Round trip time	(100 ± 25) ms	(100 ± 25) ms
	Ambient load	0% - 50% - 90%	50%
	Cost	0.50 ¢/MB	0.50 ¢/MB
Satellite	Data rate	10 Mbps	10 Mbps
	Round trip time	(200 ± 50) ms	(200 ± 50) ms
	Ambient load	0%	0%
	Cost	0.65 ¢/MB	0.65 ¢/MB

der a range of channel conditions and present results from one scenario generally representative of the trends we have observed.

In real-world deployments, cellular and Wi-Fi channels may be restricted by high traffic loads due to high demand for connectivity, whereas the satellite channels typically remain well-provisioned. We therefore vary the load over the cellular and Wi-Fi channels but maintain unloaded conditions for the satellite network. We choose a load of 50% to represent low load network conditions, 70% for moderate congestion, and 90% for extreme congestion. We excluded values that were outside of this range as values of less than 50% do not perform any differently than a system with 50% load. Furthermore, values greater than 90% introduced additional performance artifacts not related to the focus of our study.

To perform our evaluation, we developed an implementation of our system and tested it over an emulated network. We chose to implement our system, as opposed to simulating it, and tested it over emulated network conditions to achieve higher fidelity of physical and transport layer performance. Our network emulations were conducted over Emulab [110], and we used the results obtained in previous studies to accurately and realistically configure the emulation environment [27, 52, 88, 101].

Our network testbed was configured by an Emulab NS-2[4] file. The stationary agent ran on a machine with a stable connection to the Internet. It was was configured to have five 10/100 Mbps full-duplex switched Ethernet connections. The mobile agent as located on an intermittently connected machine with five Ethernet network cards. The mobile and stationary agents were connected through three of these interfaces; each interface mimics the characteristics of either the satellite, cellular, or Wi-Fi connection. We use the commands provided by Emulab to adjust the characteristics of the three network connections as listed in Table 3.1.

To extend the period of measurement and accurately characterize network throughput we chose as the basis for our experiments the task of transferring 40 MB. While 40 MB is a rather large amount of data, it was useful to demonstrate the likely behavior of the system averaged over a longer period of time. Based on our results,

[4]http://www.isi.edu/nsnam/ns/

we believe our scheme to perform equally well when used with smaller amounts of prefetched data and mini-bundle sizes.

In our evaluation, we assumed continuous connectivity for the cellular and satellite networks; therefore, we did not model interruptions caused by horizontal handoffs within either of these two communication channels. However, horizontal handoffs in the Wi-Fi network were taken into consideration. We modeled Wi-Fi disconnectivity following the suburban driving model discussed by Gass et al. [88].

In our simulations we configured data rate and delay characteristics representative of respective network channel technologies. For Wi-Fi connections we used IEEE 802.11g standard. 802.11 is widely deployed in home and municipal networks and its connection handoff characteristics are well understood [9, 88]. For the cellular connection, we considered third-generation (3G) technology recently becoming more widely deployed [27, 52]. For the satellite connection, we emulated the characteristics of Low Earth Orbit Satellites (LEOs) commonly used for commercial Internet service [101].

The metrics we use to evaluate our system are goodput and goodput-to-cost ratio.

- Combined Goodput is the rate at which data is received at the mobile agent from the combined transmissions of the underlying communication channels. It measures the average bit rate delivered to the mobile agent's application layer exclusive of all protocol overhead and data packet retransmissions. It

provides a higher-level measurement of the efficiency of our dynamic strategy. Since we deal with multimedia file transfers, combined goodput corresponds to the achieved file transfer rate over the three available parallel networks. We use this metric to measure the user-perceived performance to a particular data-to-channel allocation strategy.

- Goodput-to-cost measures system performance with respect to per transmission unit cost. It maps goodput to user cost. The higher the goodput, the greater the transmission efficiency at a particular cost. We use this ratio to evaluate the transmission choices available to the ISP to transmit data with the lowest cost.

The goal of our work is to evaluate our system with respect to goodput and goodput-to-cost ratio. In order to do so, we chose a configuration environment that enables us to study this behavior, as detailed in this section. In the next section, we present the set of results that demonstrate the performance of the ParaNets-Enabled DBS-IC static and dynamic data-to-channel allocation strategies. We focus on the changes in the performance of the evaluated strategies in response to changing network conditions. We chose values for network parameters that use realistic assumptions for the underlying network conditions as explained earlier.

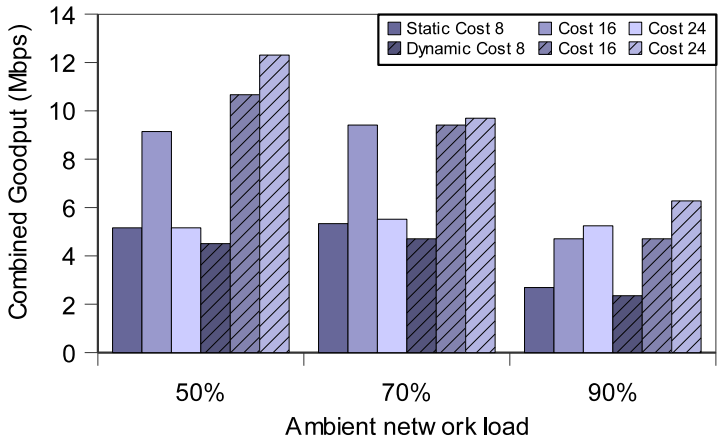


Figure 3.5: Comparative performance of dynamic and static data-to-channel allocation mechanisms.

3.5 Results

In this section, we present results sufficient to demonstrate the key findings of our work. The following results were obtained by varying network load between 50%, 70%, and 90% for the cellular and Wi-Fi channels while keeping the load at 0% on the satellite channel. We focus on the changes in the performance of the proposed schemes in response to changing network conditions. We choose these values as they reflect a realistic framework for the underlying network conditions as explained previously.

Figure 3.5 compares the performance of the static and the dynamic strategies with respect to the user-specified maximum transmission cost. Performance is measured with respect to combined goodput, which specifies the rate at which the user

is receiving data over all the parallel channels simultaneously. For each load value, we measure the average combined goodput according to the data-to-channel allocation strategy of both the static and dynamic scheduling strategies. The higher the combined goodput, the faster the user is receiving data for a given cost value and the better the achieved performance. We observe that the dynamic strategy outperforms the static strategy regardless of cost constraints. While the static strategy is sending data based on a fixed transmission schedule, the dynamic strategy determines the most efficient data-to-channel allocation mapping based on channel conditions and cost requirements. The dynamic strategy makes more accurate decisions for channel usage and outperforms the static strategy by adapting its transmission pattern to the varying network conditions.

We also observe in Figure 3.5 that, as the cost limit increases, the difference between the static and dynamic strategies becomes more pronounced. At low costs, both schemes achieve almost the same level of performance. This trend is due to the limited number of possible channel combinations available when a small cost value is imposed by the user. Increases in performance at higher allowable cost values can be attributed to the fact that the use of the highly available cellular and satellite networks for data transmission incur greater costs, but mitigate the negative effects of Wi-Fi intermittent connectivity. Alternatively, the Wi-Fi communication network is free, but periodically unavailable. At low costs, the dynamic strategy

shifts the majority of mini-bundles to be transmitted using the Wi-Fi channel, even if the wait time is long (because the channel is currently experiencing an outage). This choice is similar to the static allocation strategy, which for low cost values, essentially makes the same choice–to use the free Wi-Fi channel regardless of the time it takes to deliver all of the mini-bundles.

As we increase the load on the Wi-Fi and cellular channels from 50% to 70% and finally to 90%, the difference in the average goodput between both methods becomes less prominent. As a result of increasing the load on the cellular and Wi-Fi channels, the range of possible data-to-channel allocation strategies is narrowed to using the well-provisioned satellite network for the majority of data transmissions. As the conditions of the parallel networks are made to represent a congested network, the majority of mini-bundles will be shifted to a single channel for effective data transmission. In such a case, the difference in performance between the static and the dynamic methods becomes less pronounced.

Figure 3.6 shows the goodput-to-cost ratio for lightly-loaded channel conditions graphed on a log scale. We plot the goodput-to-cost ratio with respect to the cost that the user is willing to incur. This graph allows us to evaluate system performance with respect to per-transmission-unit cost. We observe that the goodput-to-cost ratio decreases rapidly with increases in transmission cost. This trend can be correlated with the bell-shaped behavior of the aggregate goodput-to-cost measurements as

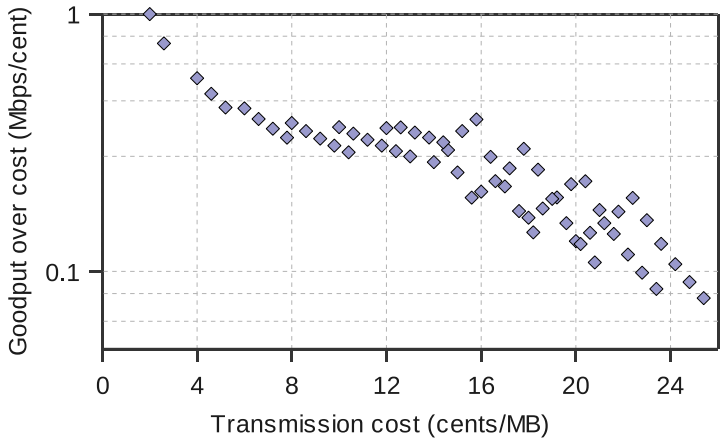


Figure 3.6: Data-to-channel allocation efficiency with respect to transmission cost.

depicted in Figure 3.3. At low costs, data is predominantly sent over the cheap Wi-Fi channel. Although goodput is low when low cost is a requirement, the low cost values will lead to an apparent tradeoff between the goodput achieved and the benefits received with respect to cost. When the user is willing to pay more, the costly alternative cellular and satellite connections will be more heavily utilized to reduce delay; the goodput-to-cost tradeoff, however, becomes less favorable.

The steep decrease in transmission cost with respect to system goodput suggests the possibility of cost savings by Internet Service Providers (ISPs). Evaluating such behavior is important in that it allows us to identify the benefits our system can provide to ISPs as well as to individual users. Through a comparison of user goodput versus the goodput-to-cost ratio, the ISP can balance user satisfaction and savings

by sending data to the user using a channel allocation strategy with slightly lower performance but much lower cost. We plan to focus on this tradeoff in future work.

3.6 Future Work

The dynamic scheduling strategy we propose in this chapter adopts a passive approach for measuring channel throughput. By sending data over a particular channel, our system is also calculating the average throughput of delivering that data and deciding on the next data-to-channel allocation mapping accordingly. Such measurements can become stale after a period of not using a particular channel. We plan to investigate the adoption of a more active approach that probes the network if a particular channel goes without being used for a long period of time.

In this work, we present one dynamic approach for channel allocation strategies. An alternative dynamic algorithm we propose would dedicate queues to each of the available network channels. The data to be sent would wait in each of these queues; when a channel is connected, we allocate a mini-bundle of data to that channel's queue. In general, such an approach would allow even more dynamic decisions to be made. The tradeoff is that bad decisions could be made in cases where a mini-bundle is allocated to a low bandwidth, slow channel. In this scenario, completion of the entire bundle transfer would be dependent on the last mini-bundle

now being sent over the slow channel. The decision would have been a bad one if a high speed channel could have completed the job much more quickly, but was not available when the original allocation to the slow queue was made. A more intelligent approach can be achieved, however, if we have some knowledge of the conditions of the network for each queue. A study of more elaborate dynamic/hybrid approaches that can combine two or more sources of information is left for future work.

Related work on TCP performance for satellite networks, particularly LEO liks, and cellular networks, particularly 3G links, have identified problems and a wide variety of solutions. A survey performed on the usage of different transport protocols in IP-based networks with satellites has identified the reasons why TCP has difficulty with satellite links [47]; other work evaluates how well end-to-end transport connections perform in a satellite environment and summarize how latency and asymmetry can impair the performance of TCP [42, 81]. Similar studies have been performed on 3G connections and have arrived to similar conclusions concerning the problems TCP faces with 3G cellular links [56]. As part of our future work, we intend to study the impact transport layer protocols on the performance of our system, and possibly identify alternative transport layer protocols that would work seamlessly within our system.

3.7 Conclusions

In this chapter, we have presented a dynamic approach to efficiently exploit the availability of heterogeneous networks in parallel. With the increasing deployment and availability of high-bandwidth wireless technology (namely, cellular and WiFi), and to support the current and future mobile data traffic growth, there is an increasing need to exploit the capacity from multiple wireless interfaces.

Our goal in this work has been to *enable multifold increases in capacity in wireless links* by providing an efficient and accurate recommendation for channel usage. Efficient usage of available channels involves reducing transmission time to the minimum possible for a given user-defined cost constraint. Our proposed dynamic strategy accomplishes this goal by adapting the data-to-channel allocation mappings to the underlying network conditions and, in so doing, providing more accurate recommendations for how to enable multifold increases in capacity in wireless physical links, through the exploitation of multiple heterogeneous wireless networks.

We build our system over our proposed architecture, ParaNets-Enabled DBS-IC, that takes advantage of the availability of multiple networks in parallel in order to increase data rates, and in so doing, to enable a perception of constant connectivity. We extend the static allocation strategy of ParaNets-enabled DBS-IC and propose adapting our channel usage scheme to the underlying network state and conditions.

We compare our adaptive strategy against this static approach under different scenarios. Our results showed that the improvement in performance varied with cost and channel conditions; the lower the cost or the more extreme the channel conditions, the less improvement the dynamic scheme shows. However, in all cases, the dynamic approach outperforms the static strategy and shows significant improvement.

We present our dynamic strategy with an emphasis on the improvement it would achieve for user-perceived performance. Through the analysis of our results, we verify that our system can also be studied with a concern for ISP-perceived performance. Through a comparison of user goodput versus the goodput-to-cost ratio, an ISP can balance user satisfaction and savings by sending data to the user using a channel allocation strategy with slightly lower performance, but much lower cost.

A dynamic approach for utilizing parallel networks is a solid step towards improving the performance of data transmissions in environments with heterogeneous connection opportunities. However, there are several areas open to future work.

Chapter 4

Online Spectrum Auctions in the TV White Spectrum

4.1 Introduction

In 2009, the FCC (Federal Communications Commission) opened up a wide range of spectrum for unlicensed usage, that was previously allocated to analog television broadcast. This meant that for devices, who until then had to operate on narrow and crowded frequencies[1], now had the opportunity to operate on a wider range of spectrum. The opportunity emerged to enable multifold increases in capacity in wireless physical links by exploiting uncrowded frequency. The challenge is to design a system that is able to efficiently redistribute or allocate this open white space spectrum to unlicensed, or secondary, users. Dynamic spectrum auction systems emerged as an effective model to do so.

[1] Specifically devices operating in the narrow WiFi frequency range, and particularly those in the 2.4GHz ISM band that suffer from external interference, such as from microwave or bluetooth.

In the context of secondary markets to redistribute white space spectrum efficiently, recent work has proposed several dynamic spectrum auction systems that periodically auction available spectrum to wireless networks producing the best economic outcomes [28, 49, 124, 125]. Using short time cycles, these auctions seek to match spectrum allocation to time-varying demand, exploiting temporal and spatial multiplexing to improve spectrum utilization and efficiency.

Running auctions periodically simplifies the auctioneer's operation, but introduces inconvenience to the bidders. For example, obtaining spectrum for periods longer than the auction cycle is cumbersome. A bidder must participate in multiple cycles and in each cycle faces the threat of being outbid and losing its spectrum usage. For the same reason, it is particularly difficult for auctioneers to choose a right auction cycle while supporting diverse spectrum demands.

Online spectrum auctions can overcome such limitation. In online auctions, bidders can request spectrum at any time. Each request includes its arrival time, monetary bid, job length (desired time duration and frequency usage), and a deadline for granting such usage. Requests are processed by the auctioneer instantaneously rather than at the start of any auction cycle. In this way, bidders can request and obtain spectrum in a genuine "on-demand" manner. This flexibility makes online auctions particularly attractive in practice.

The same flexiblility, on the other hand, introduces significant design challenges. First, the auctioneer must determine auction winners on-the-fly, without knowledge of bidders who will subsequently arrive. Such uncertainty complicates the auction design. Second, online auctions open up new vulnerabilities to selfish bidders who seek to engineer their requests to manipulate auction outcomes and gain unfair advantages. In periodic auctions, a bidder cheats only by rigging its bid and job size. In online auctions, a bidder can also cheat by falsely reporting its arrival time and deadline, referred to as "time-cheating."

The damage caused by such time-cheating is significant. Using illustrative examples, we show that by strategically engineering their arrival time, selfish bidders can obtain spectrum at much lower prices and/or block other qualified bidders. This also prevents the auctioneer from exploiting time-multiplexing to serve more bidders, and significantly degrades auction efficiency. Therefore, an effective online auction design needs to address both bid- and time-cheatings.

To resist selfish bidders, we propose *Topaz*, a truthful online spectrum auction design that discourages bidders from cheating in their bids, arrival time and deadline. To the best of our knowledge, this work is the first to address time-cheating in online spectrum auctions. The idea behind *Topaz*'s design is to combine a 3D (time, space, frequency) spectrum allocation mechanism with a *temporal-smoothed critical value* based pricing mechanism. The 3D spectrum allocation applies forward bin packing

to mitigate the uncertainty of future arrival, and at the same time enables spatial reuse and temporal multiplexing to best utilize the spectrum resource. On top of the spectrum allocation, the proposed pricing mechanism computes the price for each winner as the minimum bid required for it to win the auction. Such pricing guarantees that no bidder can improve its own utility by either rigging its bid, falsely reporting arrival/deadline, or both.

Topaz implements a "scalable" winner preemption option to address uncertainty that arises from online decisions. Normally an auction guarantees that each winner will receive its requested spectrum with no interruption. However, because the auctioneer makes on-demand decisions without knowing future arrivals, a low-bid bidder who submits its request early will block a high-bid bidder who arrives subsequently. This leads to a heavy loss in auction revenue and efficiency. Preemption can effectively mitigate such uncertainty. The auctioneer can interrupt a winner's ongoing spectrum usage and reassign spectrum to newly arrived high-bid bidders. By pinpointing high-bid bidders, preemption helps boost auction revenue [37]. This, however, is at the cost of degraded spectrum utilization (partially assigned spectrum does not offer meaningful service to its user). To explore the tradeoff between revenue and spectrum utilization, *Topaz* introduces a flexible preemption design where the auctioneer can control the aggressiveness of preemption. This design also allows us to study the tradeoff in greater detail. Our results indicate that there is an

optimal aggressiveness setting that maximizes auction revenue at a minimum loss of spectrum utilization.

We built a prototype of *Topaz* using C++, running on a standard PC with 2.4 GHz quad-core CPU and 4GB RAM. At a bidder arrival rate of 16 bidders per time unit, it takes 10ms to make an auction decision after a bidder's arrival, and 90ms to determine a winner's price after its reported deadline. This demonstrates *Topaz*'s computational efficiency.

To our best knowledge, *Topaz* is the first design to effectively resist bid- and time-cheating in online spectrum auctions. It differs from existing works on both conventional and spectrum auctions. First, *Topaz* is motivated by prior work on truthful online auctions [37, 60]. These designs, however, assume that bidders all conflict with each other and there is no spectrum reuse. In the context of spectrum auctions, *Topaz* makes an important contribution by addressing the complex interference constraints among bidders and enabling spectrum reuse to improve allocation efficiency. Second, existing work on online spectrum auctions [116, 117] only considers selfish bidders who falsely report their bids and/or job length. We show, however, that bidders can also easily manipulate auction outcomes by misreporting their arrival and/or deadline. Thus *Topaz* focuses on discouraging individual bidders from manipulating their bids and/or time reports. Finally, while prior works focus

on either preemption or no preemption, *Topaz* navigates between the two extremes while achieving the same level of truthfulness.

Limitations. The current design of *Topaz* does not address bidders who falsely report their job length. Ideally, a truthful online auction should resist all possible bidder misreports. Yet such a solution comes at a heavy cost. It is proven that to resist any type of misreport, an auction has to use agent-independent pricing (*e.g.* posted price), and suffers severe degradation in auction revenue and efficiency [37, 59, 78]. Therefore, practical designs consider a subset of misreport patterns [37, 60]. In Topaz, we restrict ourselves to consider bidders misreporting bid, arrival time, and deadline, but not job length. As we will show in Section 4.2, the incentive for bid- and time-cheating is more significant compared to misreporting job length.

4.2 Online Spectrum Auctions

As background, in this section we briefly introduce online spectrum auctions and its operation procedures. We show that compared to conventional auctions, online auctions face significant design challenges.

4.2.1 Auction Model

We consider sealed-bid online auctions where bidders, upon detecting a need for spectrum, submit requests to the auctioneer privately[2]. Each spectrum request contains the current time a (or arrival time), the job length l (the time duration), the bid b, and the deadline d for fulfilling the request. For simplicity, we assume each bidder only requests one channel. Upon receiving a request, the auctioneer then decides whether to allocate any spectrum to the bidder or to put it on hold. When a winning bidder finishes its spectrum usage, the auctioneer seeks to reassign the released spectrum to other qualified bidders with unexpired requests. The winner's price is determined at the time of its reported deadline.

In online auctions, auction decisions are triggered by bidder arrival and winner departure. Since these events occur randomly in time, the auctioneer must make decisions on-the-fly, without the knowledge of the bidders who will subsequently arrive. Therefore, the auction result is almost First-Come-First-Serve. In this case, a low-bid bidder who submits its bid earlier than a high-bid bidder could win the auction and block the high-bid bidder. This not only reduces auction revenue, but also prevents the auctioneer from assigning spectrum to those who value it the most.

[2]In order to participate in the online auction, bidders must first register with the auctioneer. This *registration phase* allows the auctioneer to identify all the potential bidders and precompute the corresponding confict graph by applying one of the existing methods [4, 77]. The conflict graph will be used when allocating the spectrum.

To prevent such revenue degradation, existing proposals apply *winner preemption* to preempt current low-bid winners to make space for newly arrived high-bid bidders [37]. While boosting auction revenue, preemption degrades spectrum utilization, damages auction credibility, and can potentially discourage bidders from participating in future auctions. It is also hard to charge any preempted winner since they receive partial spectrum usage. Thus, in some cases, the auctioneer may prefer designs without preemption.

Finally, while the ultimate goal is to process requests at any arbitrary time, in practice the auctioneer processes auction requests and makes auction decisions at fixed time units. Shorter time units offer better processing granularity and potentially better performance, but lead to higher overhead. Thus, the auctioneer determines the length of these units based on these tradeoffs. Similarly, each requested job length l follows the same time granularity.

4.2.2 Design Challenges

The flexible request format and online processing make online spectrum auctions significantly different and more difficult than conventional periodic auctions. We now present three key challenges facing online spectrum auctions.

1) Online Decisions. As discussed above, the auctioneer makes auction decisions without any knowledge of future arrivals. Such uncertainty makes the decision process challenging, particularly when deciding whether to preempt a winner.

2) 3D Spectrum Distribution. This challenge is unique to spectrum auctions where the distribution of spectrum must enable spatial reuse to improve allocation efficiency. The auctioneer needs to allocate spectrum in the time, space and frequency domains, which is highly complex given the underlying bidder interference constraints. Conventional online auction designs do not consider any spatial reuse.

3) Resisting Cheating Bidders. Bidders are selfish and seek to *engineer* their requests to control auction outcomes. In online auctions, they cheat by not only rigging their bids and job lengths, but also by falsely reporting their arrival time and deadline. The latter is particularly attractive to bidders who can tolerate some delay in spectrum usage but seek to manipulate the timing to reduce the cost of usage. As we will show in Section 4.3, such cheating can be highly effective in degrading auction fairness, efficiency and revenue.

A good online auction design needs to resist these selfish cheaters. One well-known solution is to make the auction truthful (or strategy-proof). That is, if no one can misreport its request to improve its utility, bidders will have no incentive to cheat and will report their actual spectrum requests. Resisting all types of misreports, however, is particularly difficult and costly. It has been proven that the only solution is

to use the trivial bidder-independent pricing such as posted price [37, 59, 78], which leads to severe (and unbounded) degradation in auction efficiency and revenue.

To balance the tradeoff between robustness and efficiency, the general methodology of existing works [37, 60] is to make reasonable assumptions to restrict bidder's misreport patterns. In this context, we argue that a bidder has less incentive to misreport its job length, compared to manipulating its bid, arrival time and deadline. By requesting more spectrum, a bidder risks getting a negative utility by paying more than necessary to satisfy its own request. When requesting less spectrum, the bidder's own request will not be satisfied. In either case, falsely reporting job length does not offer much utility gain. On the other hand, bid- and time-cheating present more compelling and practical attacks to online spectrum auctions. As we will show in Section 4.3, a bidder can intentionally "delay" its arrival time to avoid being charged with a high price or even block another qualified bidder while causing no harm to itself. Therefore, in this chapter, we design online spectrum auctions to resist bid- and time-based cheating.

4.3 Time-based Cheating

Before presenting our proposed auction design, in this section we discuss the behavior of time-based cheating and its impact. This allows us to understand why time-cheating is effective, which motivates our auction design.

Cheating Patterns. Bidders in online auctions arrive at different time instances, thus face different competitors. Such time-dependency allows strategic bidders to manipulate their arrival time to win the auction "cheaply." For example, a bidder X can delay its arrival time such that it competes only with low-bid bidders and wins the auction. Because most truthful auctions charge winners with the highest bid of their losing competitors [16, 35, 107, 124], X will be charged by a low-bid. Thus by cheating in time, X wins the auction easily and unfairly. Such cheating causes no harm to X as long as its reported arrival time is later than its actual arrival, and its reported deadline is earlier than its actual one, *i.e. no early arrival or late departure*. This is a practical assumption because a bidder reporting early arrival or late departure will receive spectrum outside of its usage period, degrading its own performance.

An Illustrative Example. We use an example to show the effectiveness of time-cheating. Consider a scenario where bidders A, B and D_1... D_n compete for one frequency channel. The corresponding conflict graph is shown in the upper left

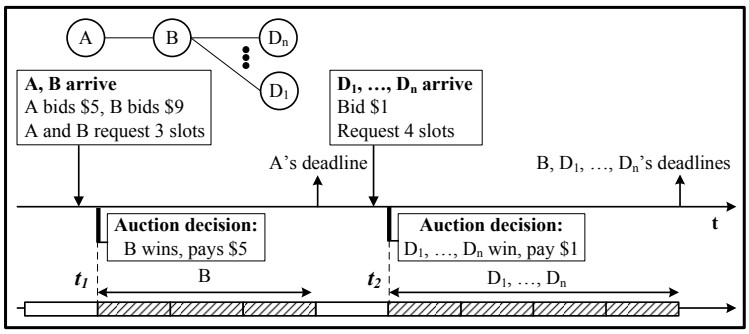


Figure 4.1: An example of online spectrum auctions. While auction events in conventional auctions occur based on fixed auction intervals, auction events in online auctions are triggered dynamically by bidders' arrivals, or changes in channel availability.

corner of the examples. Figure 4.1 plots the bidder arrival/departure and auction results when everyone behaves truthfully and reveals their true requests, assuming no preemption. In this case, two conflicting bidders A and B arrive simultaneously at t_1, and n non-conflicting bidders $D_1, ..., D_n$ arrive at t_2. In each auction event, a truthful spectrum auction design [124] is applied to determine the winners. Therefore, at time t_1, B wins the channel and is charged A's bid of \$5 (per time unit). B finishes its requested usage at t_2 and obtains a total utility = (bid-price)·job length = (9-5)·3 = 12. The auction produces a total revenue (price·job length) of \$15, an efficiency (*i.e.* sum of winners' bids) of 9·3+n(1·4), and a spectrum utilization of 3+n·4.

Now assume B strategically changes its arrival time to t_2 without changing its deadline (see Figure 4.2). Now B will compete with low-bid bidders $D_1, ..., D_n$. It also wins the auction and but pays only \$1. Thus B improves its utility, blocks the

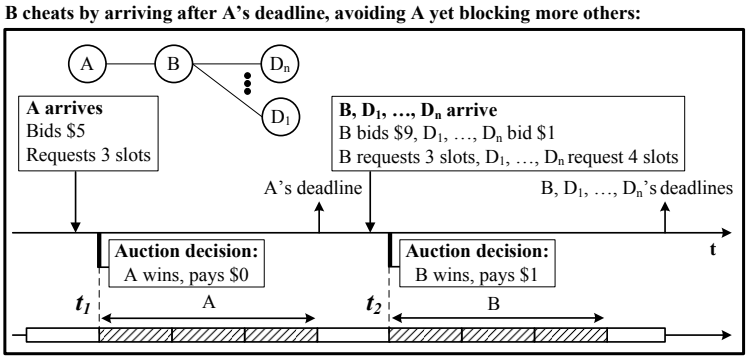


Figure 4.2: An example of bidder B's time cheating when applying existing truthful spectrum auction [124] in each slot while disabling winner preemption. Bidder B misreports its arrival time to t_2 and wins the auction cheaply. In Figure 1 it pays \$5 per slot, now it pays \$1 per slot.

n incoming bidders $D_1, ..., D_n$ at time t_2 and reduces the auction efficiency from $27 + 4n$ to 42, the revenue to \$3, and the spectrum utilization to 6.

In auctions with preemptions, time-cheating becomes easier. As shown in Figure 4.3, B can arrive even after $D_1..D_n$'s arrival and preempt these low-bid bidders $D_1, ..., D_n$. This again leads to unfair spectrum distribution and significant loss in auction efficiency.

Spectrum reuse makes time-cheating much more powerful in online spectrum auctions. As shown in the above example (Figure 4.2), B's presence at time t_2 blocks $D_1, ..., D_n$ non-conflicting bidders from using the channel simultaneously, reducing the spectrum utilization by n. Yet in conventional auctions without reuse, B can only block at most one bidder. This shows that like bid-rigging, time-cheating presents

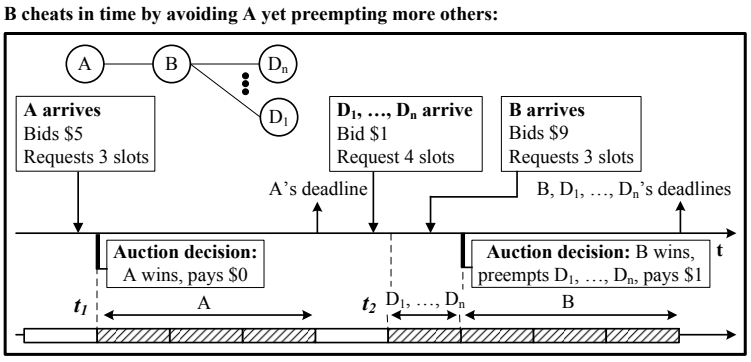


Figure 4.3: An example of bidder's time cheating when applying existing truthful spectrum auction [124] in each slot while allowing winner preemption. In this case, bidder B arrives after bidders $D_1, ..., D_n$, but still blocks them via winner preemption and wins the auction by paying a much lower price of $1 rather than $5.

a critical threat to online spectrum auctions. To build a practical and deployable system, we must design auction rules to resist both bid- and time-cheating.

4.4 Resisting Bid- and Time-Cheating

We propose *Topaz*, a truthful design for online spectrum auctions. *Topaz* effectively discourages bidders from cheating in both time and bid by enforcing the following generalized truthfulness property:

Definition 1. Let v_i, a_i and d_i represent bidder i's true evaluation, arrival time and deadline. An online auction is (a, d, v)-truthful if and only if no bidder i can improve

Table 4.1: Notations.

Γ_t	Auction event occurred at time t
a_i	Bidder i's true arrival time, its reported arrival time a'_i can only be $a'_i \geq a_i$
d_i	Bidder i's true deadline, its reported deadline d'_i can only be $d'_i \leq d_i$
l_i	Number of contiguous slots on one channel requested by bidder i
v_i	The benefit bidder i obtains for per-slot usage of one channel if it finishes its task
b_i	The maximal per-unit price a bidder i is willing to pay for the spectrum, if its request is satisfied
p_i	The per-slot price charged to i if it finishes its task
u_i	Bidder i's utility, calculated as $l_i \cdot (v_i - p_i)$ if it wins the auction, otherwise 0

its utility by bidding $b_i \neq v_i$, or falsely reporting its arrival time $a'_i > a_i$, or deadline $d'_i < d_i$, or any combination of them[3].

We now describe *Topaz* in detail. We first present the general methodology for enforcing the (a, d, v)-truthfulness, and then describe *Topaz*'s detailed procedure and an illustrative example. Table 4.1 lists the notations used in our design.

4.4.1 Design Methodology

Enforcing truthfulness requires significant efforts in both allocation and pricing. The general guideline (in periodic auctions) is to make the spectrum allocation monotonic and to use critical-value based pricing, charging each winner by the minimum bid required to win the auction [124]. This has been shown to effectively prevent bid-rigging. In online auctions where bidders can manipulate the arrival and

[3]As discussed in Section 4.3, we assume that bidders do not cheat by reporting early arrival ($a'_i < a_i$) or late departure ($d'_i > d_i$), because these disrupt its own spectrum usage.

deadline, however, we must now extend the original concept to resist cheating in both time and bid.

To achieve the (a, d, v)-truthfulness, we introduce two requirements: *monotonic allocation* and *temporal-smoothed critical value-based pricing*. The first requirement ensures the existence of a critical value for each bidder i such that i can only win the auction by bidding higher than this value. The second requirement computes the critical value by taking into account the time dependency across subsequent auction events, diminishing the gain of any bid and/or time cheating. Finally, we introduce a *scalable preemption* feature where the auctioneer controls the aggressiveness of auction preemption to balance auction revenue and spectrum utilization.

Monotonic Allocation. The allocation needs to be monotonic in bids. That is, given the arrival and deadline constraint of bidder i, the higher i bids, the more likely i wins.

Definition 2. The allocation is monotonic if the following holds: for each auction winner w, if w wins the auction by bidding (a_w, d_w, b_w), then w still wins by bidding (a_w, d_w, b'_w) if $b'_w \geq b_w$, assuming all other requests remain the same.

The monotonicity is essential to guarantee the existence of a *critical value* $\eta_i(t)$ for each bidder i in any auction event Γ_t. The critical value $\eta_i(t)$ is defined as the

value at which, if bidder i's bid $b_i \geq \eta_i(t)$, then i will win the auction Γ_t. This value will be used to price i if it wins.

Topaz achieves monotonicity by allocating bidders in a bid-dependent manner. In each auction event triggered by a bidder arrival or winner departure, the auctioneer sorts the bids of qualified bidders in a non-increasing order and allocates spectrum to them sequentially. To enable spectrum reuse, *Topaz* uses the 3D bin-packing algorithm to address the interference constraints among bidders.

Temporal-Smoothed Critical Value based Pricing. In online spectrum auctions, a bidder's critical value depends not only on other bidders' bids, but also on the time constraints. *Topaz* captures this time dependency using the *temporal smoothed critical value*. If an auction winner i reports its arrival time and deadline as (a_i', d_i') and its job length as l_i, we calculate for each $t \in [a_i', d_i' - l_i]$ the minimum bid $\rho_i(t)$ that i must bid to win the slots $[t, t+l_i-1]$. A winner i's temporal-smoothed critical value (and its per-slot price) is

$$p_i = \min_{t \in [a_i', d_i' - l_i]} \rho_i(t). \tag{4.1}$$

Charging i by p_i ensures the (a, d, v)-truthfulness by removing the time dependency. This is because, under the assumption of no early arrival or late departure, we have $[a_i', d_i') \subseteq [a_i, d_i)$, thus $\min_{t \in [a_i', d_i' - l_i]} \rho_i(t) \geq \min_{t \in [a_i, d_i - l_i]} \rho_i(t)$. This means that the price charged to i when it cheats is no less than that when it reports truthfully.

This enforcement diminishes gain from any bid and/or time-cheating. In summary, the total price charged to a winner i is $p_i \cdot l_i$. If a bidder i does not fully receive its requested spectrum before its deadline d'_i, $p_i = 0$.

Scalable Auction Preemption. When a newly arrived bidder places a bid higher than that of existing winners, the auctioneer can choose to preempt existing winners to make up the price difference. On the other hand, since preempted bidders are not charged for their partial spectrum usage, preemption does not necessarily translate to gain in auction revenue. Yet it does lead to loss in *effective* spectrum utilization since the allocated spectrum does not fulfill bidder request. Intuitively, the auctioneer should preempt a winner only if the newly arrived (and conflicting) bidder offers a significantly higher bid.

To control the preemption frequency, *Topaz* introduces a bid adjustment procedure, priortizing ongoing winners by artificially raising their bids. For a winning bidder i, who requests l_i slots and has used one spectrum channel for l'_i slots from time $t - l'_i + 1$ to t, *Topaz* will treat i's bid as $\hat{b_i}(t)$ when ranking bidders at time t:

$$\hat{b_i}(t) = b_i \cdot f^{\varphi_i} \geq b_i, \tag{4.2}$$

where $\varphi_i = l'_i/l_i$ represents i's progress at time t, and $f \geq 1$ is the factor reflecting the auctioneer's preemption aggressiveness. $f = 1$ maps to the conventional preemption model. By increasing f, the auctioneer adds more protection to allocated

winners, leading to a smaller probability of preemption. When $f \to \infty$, $\hat{b}_i(t) = \infty$, the allocated bidder i will not be preempted but will receive continuous spectrum usage. In this case, the auctioneer disables preemption completely.

4.4.2 Detailed Design

Driven by the above allocation and pricing methodology, we now describe *Topaz* in detail. We focus on cases where preemption is allowed but its aggressiveness is controlled via f. *Topaz* without preemption is a special case with $f = \infty$.

Allocation. In online auctions, the allocation decision occurs at *critical time points*, when a winner finishes its spectrum usage and releases an occupied spectrum channel, or when a new bidder arrives and submits its request. At each critical time point τ, *Topaz* sorts the qualified bidders' bids in a non-increasing order, and applies a 3D bin packing method to allocate the spectrum to bidders sequentially following their orders. For a candidate bidder i, *Topaz* "packs" the bidder's spectrum allocation forward in the next time slot using the lowest indexed channel that is available to i, *i.e.* not occupied by any bidder i's conflicting peers. Such forward packing enables spatial reuse while using current available channels to serve as many bidders as possible. While a similar concept is used by most online scheduling algorithms [33], *Topaz* extends it to cover the time, frequency and spatial domains.

Because *Topaz* allows preemption, the winners currently using the spectrum will also be considered in the above allocation procedure. The winners' bids will be raised according to (4.2). We note that, by allowing preemption, a winner's allocated spectrum usage becomes "temporary." In *Topaz*, we assume that when a bidder i wins the auction at time t, its assigned spectrum usage is only guaranteed for the current slot $[t, t+1]$, and it faces the danger of being preempted in future time slots. Preempted bidders can be re-allocated before their deadlines, but each re-allocation must cover the entire request l_i as if the winner has not received any spectrum. This is because we assume each spectrum request is non-preemptive and must be served continuously in time.

Algorithm 1 shows the step-by-step allocation procedure at a critical time τ, assuming initially no channel is allocated for the slot τ. The function $Used(\mathcal{A}, i, \tau)$ returns the number of continuous slots that i has received before τ, $Top(B)$ returns the bidder with the highest bid in B, $NC(i, G, \tau)$ returns the number of channels in the current slot τ that have been allocated to i's conflicting peers defined by the conflict graph G, *Allocate* $(i, \tau, \mathcal{A})$ allocates the current slot τ of the lowest indexed channel available to bidder i, and finally $Preempt(i, \tau, \mathcal{A})$ preempts i if i is allocated at $(\tau - 1)$.

Pricing. Pricing a winner i includes two steps. First, *Topaz* calculates, for each $t \in [a'_i, d'_i - l_i]$, the minimum bid $\rho_i(t)$ required for i to win l_i contiguous slots

Algorithm 1 *Topaz*-Alloc($\tau, B, \mathcal{A}, f, G, K$)

Input: 1) critical time τ; 2) bids B; 3) current allocation $\mathcal{A}$; 4)

preemption preference factor f; 5) conflict graph G; 6) K channels

1: $\hat{B} = \emptyset$

2: **for** $b_i \in B$ **do**

3: $\varphi_i = \text{Used}(\mathcal{A}, i, \tau)/l_i$

4: $\hat{b_i} = b_i \cdot f^{\varphi_i}$

5: $\hat{B} = \hat{B} \cup \{\hat{b_i}\}$

6: **end for**

7: **while** ($\hat{B} \neq \emptyset$) **do**

8: $i = \text{Top}(\hat{B})$

9: **if** $\text{NC}(i, G, \tau) < K$ **then**

10: Allocate$(i, \tau, \mathcal{A})$

11: **else if** i was using a channel at $(\tau - 1)$ **then**

12: Preempt$(i, \tau, \mathcal{A})$

13: **end if**

14: $\hat{B} = \hat{B} \setminus \{\hat{b_i}\}$

15: **end while**

starting from t. We hereby refer to $\rho_i(t)$ as the *interval price* of i within $[t, t+l_i-1]$. When preemption is allowed, $\rho_i(t)$ needs to be high enough so that winner i would not be preempted at any point within $[t, t+l_i-1]$. This requires us to compute, for each slot t' within $[t, t+l_i-1]$, the minimum bid required for i to win this slot. Let this value be $\eta_i(t')$, $t' \in [t, t+l_i-1]$. Since i's bid will be raised at t' by $f^{(t'-t)/l_i}$, we need to divide $\eta_i(t')$ by the same factor to get the minimum required value for i's original bid. Then the interval price is the maximum of all the qualified slots:

$$\rho_i(t) = \max_{t' \in [t, t+l_i-1]} \frac{\eta_i(t')}{f^{(t'-t)/l_i}}. \tag{4.3}$$

When no preemption is allowed ($f = \infty$), this reduces to the minimum bid for i to win the first slot t, $\rho_i(t) = \eta_i(t)$.

Second, *Topaz* applies the time-smoothing in (4.2) by checking all possible intervals starting in $[a_i', d_i' - l_i]$ and charging i with the minimal interval price among all possible intervals. Thus i's final per-slot price is:

$$p_i = \min_{t \in [a_i', d_i' - l_i]} \{ \max_{t' \in [t, t+l_i-1]} \frac{\eta_i(t')}{f^{(t'-t)/l_i}} \}. \tag{4.4}$$

Algorithm 2 lists the detailed steps of computing p_i for bidder i at its reported deadline d_i'. $UnfinishedBidder(\mathcal{A}, B, t)$ returns the set of bidders who have not finished their tasks till time t. The function *CalCriticalVal*(B, i, G, C, t) returns the minimum bid for bidder i to win the current slot t given others' bids B, the conflict

graph G, and the set C of currently available channels. The minimum bid calculation is the same as the critical value calculation described in [124].

An Illustrative Example. Consider the example in Figure 4.1. For simplicity, we change the setting to assume that bidder B's reported deadline is $d'_B = t_1 + 4$. Since B arrives at time t_1, there are two possible intervals of size 3-slots B can win before its reported deadline, which are $\Delta_1 = [t_1, t_1 + 2]$ and $\Delta_2 = [t_1 + 1, t_1 + 3]$. First, for Δ_1, B needs to beat A's bid to win each slot, meaning that the critical value for each slot is $b_A = 5$. Hence the interval price of Δ_1 for B is $\rho_B(t_1) = \max\{5, 5/8^{1/3}, 5/8^{2/3}\} = 5$. Second, if B arrives at $t_1 + 1$, A would have been allocated with slot t_1. Thus for B to win the interval Δ_2, B needs to preempt A by beating A's raised bid at time $t_1 + 1$, which is $5 \cdot 8^{1/3} = 10$. Hence the interval price of Δ_2 is $\rho_B(t_1 + 1) = \max\{5 \cdot 8^{1/3}, 5/8^{1/3}, 1/8^{2/3}\} = 10$. Therefore, B's final per-slot price is $p_B = \min\{\rho_B(t_1), \rho_B(t_1 + 1)\} = 5$, and its total price is $5 \times 3 = 15$.

4.5 Theoretical Analysis

In this section, we prove that *Topaz* achieves the (a, d, v)-truthfulness (Definition 1) for all values of f. We also discuss the performance bound on *Topaz*'s revenue and efficiency.

Algorithm 2 *Topaz*-Pricing($i, B, \mathcal{A}, f, G, C$)

Input: 1) bidder i; 2) bids B; 3) current allocation $\mathcal{A}$; 4) preemption

preference factor f; 5) conflict graph G; 6) available channels C

if Used$(\mathcal{A}, i, d'_i) < l_i$ **then**

  $p_i = 0$

  Return

end if

for $t \in [a'_i, d'_i - l_i]$ **do**

  $\hat{B} = \emptyset$

  $list$ = UnfinishedBidder($\mathcal{A}, B, t$)

  for $x \in (list \setminus \{i\})$ **do**

    $\varphi_x =$ Used$(\mathcal{A}, x, t)/l_x$

    $\hat{b_x} = b_x \cdot f^{\varphi_x}$

    $\hat{B} = \hat{B} \cup \{\hat{b_x}\}$

  end for

  $\eta_i(t) =$ CalCriticalVal$(\hat{B}, i, G, C, t)$

end for

for $t \in [a'_i, d'_i - l_i]$ **do**

  $\rho_i(t) = \max\{\frac{\eta_i(t')}{f^{(t'-t)/l_i}} | t' \in [t, t + l_i - 1]\}$

end for

$p_i = \min\{\rho_i(t) | t \in [a'_i, d'_i - l_i]\}$

4.5.1 Proof of Truthfulness

We first prove that the allocation is monotonic, and each winner is charged with the minimum bid required to win the auction. We then prove *Topaz* is (a, d, v)-truthful by showing a bidder cannot improve its utility by manipulating its bid and time report.

Lemma 1. If bidder i wins in slot t with bid b_i, it can also win by bidding $b'_i \geq b_i$, assuming that all other requests remain the same.

Proof. As shown in Algorithm 1, *Topaz* allocates bidders in a non-increasing order of bids or enlarged bids (Equ. (4.2)). For a single bidder i, since $f^{\varphi_i} \geq 1$, when i bids $b'_i \geq b_i$, we still have $b'_i * f^{\varphi_i} \geq b_i * f^{\varphi_i}$. Hence i will be ranked higher with b'_i in Algorithm 1, meaning that less bidders are considered for allocation before allocating i, compared to the case when i bids b_i. Since Algorithm 1 does not deallocate bidders, the number of available channels will only decrease as more bidders are allocated. Assume i is allocated with channel m at t_i when bidding b_i, channel m must also be available for i when it bids $b'_i \geq b_i$. Hence i can also win by bidding b'_i. This completes our proof. □

Lemma 2. $\rho_i(t)$ in Equ. (4.4) is the minimum i needs to bid in order to win the requested contiguous slots at $[t, t + l_i - 1]$.

Proof. To prove this claim, we need to show that 1) if bidder i bids less than $\rho_i(t)$ in Equ. (4.4), then i will not win l_i contiguous slots from t, and 2) if i bids no less than $\rho_i(t)$, then i will win l_i contiguous slots starting from t.

First, when $b_i < \rho_i(t)$, there must exist a critical time point $\tau \in [t, t+l_i-1]$ such that $b_i < \eta_i(\tau)/f^{(\tau-t)/l_i}$. Then, we have i's elevated bid $\hat{b_i}(\tau) = b_i \cdot f^{(\tau-t)/l_i} < \eta_i(\tau)$. Since the auctioneer will reconsider the allocation at τ, and $\eta_i(\tau)$ is the minimal amount i needs to bid in order to win the unit at time τ, i must be preempted by the auctioneer, and hence the total number of contiguous units in $[t, t+l_i-1]$ must be less than l_i. Second, when $b_i \geq \rho_i$, we must have $b_i \geq \eta_i(\tau)/f^{(\tau-t)/l_i}$ for each critical time point $\tau \in [t, t+l_i-1]$. Hence, i must be able to obtain l_i continuous units from time t. □

Theorem 1. *Topaz* is (a, d, v)-truthful.

Proof. To show (a, d, v)-truthfulness, we prove that any bidder i cannot improve its utility by 1) setting its bid $b_i \neq v_i$, 2) manipulating its arrival time $a'_i > a_i$ or deadline $d'_i < d_i$, or via any combination of them.

First, we show that i cannot benefit from rigging its bid. Without loss of generality, consider a single auction event $\Gamma(\tau)$ at critical time point τ. Lemma 1 ensures the existence of a critical value such that i wins only if it bids no less than this value.

Based on this, the proof of truthfulness follows similar lines to an existing solution in [124]. We do not discuss the proof due to the limited space.

Second, we show that i cannot improve its utility by setting its arrival time $a'_i > a_i$ or deadline $d'_i < d_i$. Let u'_i, u_i be i's utilities with arrival and deadline (a'_i, d'_i) and (a_i, d_i) respectively, and let p'_i, p_i be the prices i needs to pay in each case. We show that $u'_i \leq u_i$ in all cases:

- *i loses in both cases:* $u'_i = u_i = 0$, so our claim holds.
- *i wins with (a'_i, d'_i) and loses with (a_i, d_i):* From Table 4.1, we know that $a'_i \geq a_i$, $d'_i \leq d_i$, and hence $[a'_i, d'_i) \subseteq [a_i, d_i)$. Therefore, this case cannot occur.
- *i loses with (a'_i, d'_i) and wins with (a_i, d_i):* By Lemma 1, we know that $p_i \leq b_i = v_i$, hence $u_i = v_i - p_i \geq 0$. Since $u'_i = 0$, our claim holds.
- *i wins in both cases:* By Lemma 2, p_i is the minimal price that i needs to pay for winning requested contiguous units within $[a_i, d_i)$, and p'_i is the minimal price for winning within $[a'_i, d'_i)$. Since $[a'_i, d'_i) \subseteq [a_i, d_i)$, we have $p_i \leq p'_i$, then $u_i = v_i - p_i \geq v_i - p'_i = u'_i$.

By showing that i cannot improve its utility by setting either $b_i \neq v_i$, or $a'_i > a_i$ or $d'_i < d_i$, we prove that *Topaz* is (a, d, v)-truthful. □

4.5.2 Efficiency and Revenue Bound

To evaluate *Topaz* by its auction efficiency and revenue, we compare it against the *offline Vickery* mechanism which makes auction decisions with knowledge of all the bidders who subsequently arrive into the system. Providing an upper bound on the auction performance, this offline solution is typically used to evaluate online mechanisms [37]. The auction efficiency is defined as the sum of winners' bids, and the auction revenue is defined as the sum of winners' charges.

The performance of *Topaz* depends heavily on the underlying bidder interference constraints, or the conflict graph. When bidders conflict with each other, *i.e.* the conflict graph is a complete graph, we can use existing results to verify *Topaz*'s efficiency and revenue bound. Following the same method in [37], we can show that when bidders have the same job length, and all conflict with each other, *Topaz* with $f = 2$ is 5-competitive in terms of auction efficiency. On the other hand, even in this simple scenario, [37] has shown that there is no deterministic truthful mechanism whose revenue is constant-competitive with that of the offline VCG mechanism.

Unfortunately, we are unable to derive any bound under general conflict graph, and thus delay this to a future effort. Instead, we use simulations to examine *Topaz*'s revenue and efficiency trends under sample bidding and arrival behaviors.

Table 4.2: Implementation Configuration.

Auction Parameter	[Range], Nominal	**Bidder Behavior**	Model
# of channels	[1,10], 5	Arrival	Poisson, Uniform random
Slots per auction	[1,200], 50	Bid	Uniform random, Beta
# of bidders	[1,800], 600	Request length	Uniform, Random bounded
Factor of preemption preference f	[1,100] & ∞, 2	Channel requested	1 channel per bidder

4.6 Simulation Results

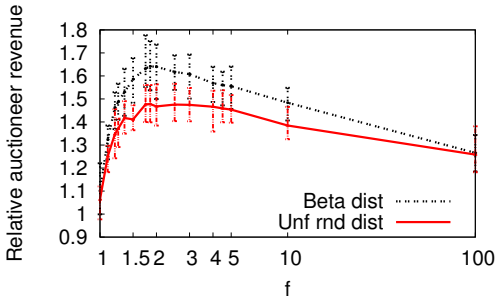
In this section we use simulation experiments to evaluate *Topaz* under illustrative bid distributions and arrival models. We focus on examining how to configure *Topaz* if we are to use preemption and compare our designs with different f settings.We conclude with a complexity analysis to determine how feasible *Topaz* is for real-time, online spectrum auctions. We did not compare *Topaz* to existing works because no prior solutions have achieved the generalized truthfulness in an online spectrum auction setting; thus, any such comparison is unfair. Instead, we examine *Topaz* under varying conditions, in order to isolate the conditions that capture the important behavioral patterns of our design.

We implement *Topaz* in C++ using the configuration parameters listed in Table 4.2. While modeling the bidding behavior itself is an open problem in the field of economics, we choose a set of configurations that best represent typical online spectrum auctions. After proving *Topaz*'s truthfulness under any bids, our simulation study is to examine its behavior under varying conditions. Because *Topaz* ensures truthfulness by giving bidders no incentive to cheat, we assume bidders bid

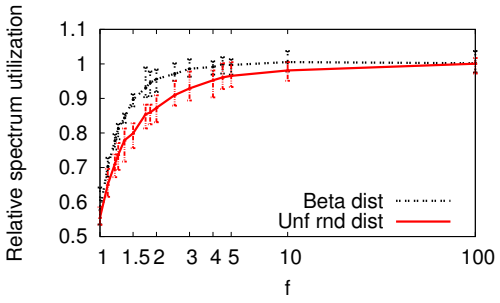
by their true valuation and arrival/deadline. We consider two models, used in recent auction studies [125, 126], that best represent how users value common goods: uniformly random distribution in a range or beta distribution where the values are concentrated near a common value. We have tried many different configurations and the results reveal similar trends. Finally, we use the same method as [124] to produce bidder interference constraints. Our results again reveal similar trends. In this section we will show the most representative results. We average the results over 10 runs.

4.6.1 To Preempt or Not?

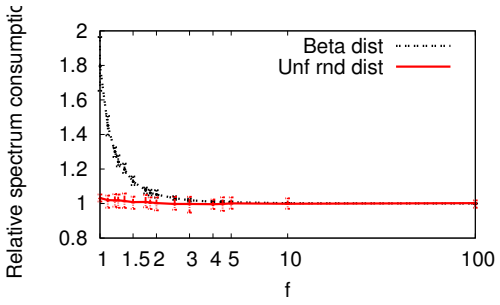
We first study the impact of f which determines the preemption aggressiveness . Intuitively, a small f will lead to frequent preemptions that waste spectrum (particularly because preempted bidders are not charged). Auctions with a large f, on the other hand, will likely be trapped by low bids and receive less revenue. Therefore, we need to carefully choose f to target high bidders without frequent preemptions. Figure 4.4 illustrates the impact of f in terms of *Topaz*'s auction revenue, spectrum utilization (the total amount of spectrum assigned to non-preempted winners), and the amount of spectrum consumed by both preempted and non-preempted winners. We normalize each metric by that of *Topaz* without preemption (WOP), *i.e.* $f = \infty$.



(a) Auctioneer revenue.



(b) Spectrum utilization.



(c) Spectrum overhead.

Figure 4.4: Impact of different values of f on auctioneer revenue, effective spectrum utilization (assigned to non-preempted winners), as well as total spectrum consumption (assigned to all winners), respectively. We divide the values by those of WOP where $f = \infty$. We run the design for two bid distribution models.

Figure 4.4(a) examines the auction revenue. At first, the revenue increases rapidly because increasing f reduces preemption frequency and improves bidders' chances of satisfying their spectrum requests. As f increases further and preemption becomes harder, *Topaz* becomes trapped by low bids, and, consequentially, its revenue falls and slowly converges to that of WOP. At $f = 2$, the auction does reach a balance, which appears to be the optimal point in terms of the revenue.

Spectrum utilization, as shown in Figure 4.4(b), displays a similar exponential increase as in Figure 4.4(a), but the increase gradually levels out and converges to 1. It is clear that high values of f maximize spectrum utilization as they approach the upper bound set by WOP. For a revenue-optimal value at $f = 2$, we sacrifice 5-15% of the maximum attainable spectrum utilization. Auction efficiency follows a similar trend, therefore we exclude its results. We deduce that there is a trade-off between revenue, and auction efficiency/spectrum utilization. For an auctioneer, who aims to maximize revenue, the optimal value of f would be 2 at a relatively low cost in terms of efficiency and spectrum utilization.

In Figure 4.4(c), we look at relative spectrum consumed, a measure of spectrum wasted. For *Topaz* with $f = 1$, 80% of the spectrum is wasted. As f increases, the preemption frequency reduces and the waste rapidly decreases and converges to 0. At $f = 2$, the spectrum waste is reduced to 0.01-0.05%, which is a negligible cost to maximize auction revenue.

We suspect that the optimal value of $f = 2$, observed in Figure 4.4(a), is affected by the bid scheme. Therefore, we run our allocations using two different bid distribution models: random bids with a minimum, maximum, and average values of 50, 100, and 150, respectively, and a beta distribution with $\alpha = 5$ and $\beta = 5$. We can see that $f = 2$ is in fact the optimal point and that the behavior of our evaluation metrics due to f is similar regardless of the bid distribution model; we leave the task of analytically finding the best f value to future work. Another key finding is that all possible values for f result in a positive gain (up to 65%) over WOP, confirming that some preemption is always beneficial towards revenue.

Preemption Frequency To understand the impact on bidders, Table 4.3 shows the distribution of the number of preemptions each winner receives. *Topaz* with $f = 2$ remains relatively reasonable since 72% of allocated bidders do not experience any preemption. Only 7% of bidders are preempted twice and 3% are preempted more than twice.

Summary. These results also show that *Topaz* with *proper* preemption ($f = 2$) significantly outperforms *Topaz* without preemption in terms of auction revenue, at a small loss in spectrum utilization and auction efficiency. Although disabling preemption provides security to winners, it suffers much lower revenue. Thus disabling preemption will be useful in scenarios where stability and consistency are priori-

Table 4.3: Distribution of the number of preemptions per bidder

# of preemptions	0	1	2	3	4
Topaz ($f = 2$)	72%	18%	7%	2%	1%

tized while proper preemption is more flexible and useful in scenarios that favor high revenue returns.

4.6.2 Auction Complexity

We examine *Topaz*'s run-time performance in Table 4.4, including the maximum time required for the auctioneer to determine the allocations at a time t and the time to determine the price for a winner after a successful spectrum allocation. Because the run-time depends on the bidder arrival rate, we consider 100, 400 and 800 bidders who arrive sequentially across 50 time units. We ran our experiments under different arrival models and observed similar behavior. These results demonstrate the feasibility of running *Topaz* in practice.

Both allocation and pricing times scale with the arrival rate. However, *Topaz* ($f = 2$) and *Topaz* ($f = \infty$) require minimum computation time. In particular, the allocation time of *Topaz* without preemption increases almost linearly with the number of bidders and reaches a maximum of 10ms. This shows that *Topaz* can quickly react to dynamic bidder arrivals. The pricing time, on the other hand, increases linearly for *Topaz* with preemption; this effect is due to re-running the allocation to

Table 4.4: *Topaz*'s Processing Time for Allocation and Pricing (running on a PC with 2.4GHz quad-core CPU and 4GB RAM).

Allocation time (ms)		
# of bidders in 50 time units	*Topaz* ($f = \infty$)	*Topaz* ($f = 2$)
100	0.0	2.0
400	6.0	10.0
800	10.0	10.0
Pricing time (ms)		
100	0.06	6.20
400	0.50	42.41
800	0.87	89.79

determine pricing for each winner, and the number of winners increases with the number of bidders. When disabling preemption, the pricing time becomes negligible.

4.7 Related Work

There has been rich literature on online mechanism designs and spectrum auctions. [37] proposed series of online auction designs to combat bidder's cheating on bid, job length, arrival time and deadline. The authors have proven the performance bounds for these designs towards maximizing revenue and auction efficiency. Yet unlike *Topaz*, these designs assume bidders all conflict with each other and hence cannot exploit the spatial reuse when directly applied in spectrum auctions. A recent work in [116] proposed online auction designs with spectrum reuse and preemption. These designs, however, do not address cheating on arrival and deadline. The work in [124, 125] designed truthful auctions with spectrum reuse by using periodic auc-

tions. In contrast, *Topaz* judiciously integrates online allocation and pricing with flexible winner preemption, and succeeds to resist bidder's cheating on bid as well as arrival/deadline in online spectrum auctions.

As another line of related works, bin packing and scheduling algorithms have been widely applied in various applications with deadline constraints [14, 26, 44, 102]. Several effective scheduling algorithms have been proposed to minimize task completion time [68,69] or consider task types and deadlines [51,65]. Alternatively, we focus on packing requests in online systems. Given the complex interference constraints, finding an optimal packing algorithm is NP-hard. Thus we focus on fast greedy algorithms that can be deployed in practice and yet offer the same level of auction truthfulness.

4.8 Conclusion and Future Work

Our work proposes an effective approach to distributing white space spectrum to users. In so doing, we have provided a spectrum access approach that allows wireless users, who until then had to operate on narrow on crowded frequency, the opportunity to operate on a wider, less congested frequency range. White space spetrum is a direction in wireless systems that enables and allows multifold increases

in capacity in wireless physical links through efficient and effective usage of open spectrum.

In this chapter, we consider online spectrum auctions where bidders, or users, arrive and depart dynamically. This network model is characteristic of wireless bandwidth usage, where users' demand for bandwidth is time-varying [124]. We propose *Topaz*, a truthful online spectrum auction design. *Topaz* makes an important contribution by enabling spectrum reuse and discouraging bidders from cheating in bids and time reports.

We point out several directions as future work.

Requests without contiguity. In this work, we assume bidders request contiguous allocations and thus do not charge winners receiving partial spectrum usage. In practice, some bidders can accept non-contiguous allocations. This reduces the impact of preemption on bidder utility. It is interesting to explore auction rules that enforce truthfulness in such auction systems and to study the impact of preemption.

Misreporting request length. Another direction is to consider when bidders misreport their request time length l, to a limited extent, since fully addressing all types of misreport is only achievable via bid-independent designs.

Bidder-collusion. Topaz focuses on addressing individual bidder cheating without collusion. In practice, bidders can form groups and manipulate their requests together [126]. Addressing collusion, however, requires more strict rules.

Chapter 5

Channel Width Adaptation

5.1 Introduction

Another wireless technology that has provided opportunities to exploit higher bandwidth in wireless networks is IEEE 802.11 WiFi. WiFi has witnessed exponential improvements in data rates, supporting up to 300Mbps with the 802.11n standard and gigabit Ethernet speeds with 802.11ac. This exponential increase in data rate has been achieved through a number of technologies that have been added to the WiFi chipset, namely wider channel widths and higher-order Multiple-Input Multiple-Ouput (MIMO) smart-antenna technology. In this chapter through Chapter 7, we study, develop, and propose methods of exploiting the high speed links enabled through WiFi technologies. In this chapter, we focus our efforts on enabling the bandwidth increases from wider channel widths through channel bonding

in 802.11n. We believe our work will also apply to the 802.11ac standard that allows up to 160MHz bonding channels in the 5GHz band.

With the wide deployment of the IEEE 802.11n standard and with 802.11ac, WLANs now have the option to operate over wider channels that achieve higher capacity. The standardized 802.11n technology supports up to 40MHz channels through channel bonding, where two 20MHz channels are combined into a single 40MHz channel. Although transmissions over 40MHz channels should provide advantages over 20MHz channels, performance benefits are largely influenced by the adopted antenna technology. With the incorporation of MIMO smart-antenna technology in 802.11n devices, problems faced by traditional Single-Input Single-Output (SISO) systems from channel bonding [15,84] can now be mitigated [29,39]. MIMO technologies in 802.11n promise new potential for channel bonding and higher transmission rates.

Wider bandwidths are also faced with challenges. The IEEE 802.11n standard imposes a fixed maximum transmission power on devices. By doubling the channel width, SNR is effectively decreased by 3dB [7], and thus, reception errors increase [85]. Furthermore, wider bandwidths are more likely to suffer from frequency selective fading. A 40MHz channel, therefore, not only requires a stronger transmission power to achieve the same SNR but also a higher SNR to provide the same PER. That is, transmissions using channel bonding require a slightly stronger sig-

nal strength to provide the same reliability as that of a single 20MHz channel. This tradeoff between higher transmission rates and susceptibility to interference must be carefully understood in order to improve performance. The 802.11n standard itself gives no guidelines or recommendations on how to benefit from channel bonding [1].

Previous experimental studies on 802.11n provide valuable insights into 802.11n features [7, 82, 85, 99], but fall short in effectively characterizing the opportunities for channel bonding in real-world WLAN settings, where interfering links co-exist. Furthermore, most existing work operates within the 2.4GHz ISM band [82, 85, 99], where channel constraints are too tight to effectively gauge the performance of channel bonding. In fact, it was shown that channel bonding in the 2.4GHz range poses more harm than benefits [15, 99, 103]. There is therefore a clear need to evaluate the behavior of and opportunities for channel bonding under a broader range of circumstances, where the benefits of channel bonding can truly be exploited: the 5GHz range. In fact, the emerging 802.11ac standard operates only on the 5GHz band for this very reason.

We set out to identify the usage conditions for channel bonding in 802.11n WLANs [20]. These usage terms allow for intelligent channel bonding decisions and efficient utilization of available spectrum. To this end, we first characterize the behavior of channel bonding through experimental studies. Experiments were performed in the 5GHz frequency range over a stationary 802.11n testbed deployed in

a semi-open office environment. These experiments demonstrated the impact of network conditions and interference patterns on throughput performance with channel bonding. As a result, we identify channel bonding opportunities in WLAN environments to achieve higher transmission rates.

From our experiments, we discover that most of the naïve channel bonding decisions in fact degrade performance. Intelligent channel bonding decisions require knowledge of not only a link's signal quality, but also of the strength of neighboring link's transmissions, their channel distance, their physical rates, and their traffic load. For example, transmissions from neighboring links with strong signal strengths to each other could lead to interference from channel leakage from transmissions on non-overlapping yet consecutive channels. Similarly, for links in carrier sensing (CS) range that are operating on overlapping channels, a link with a low physical rate could degrade performance of the other links. We identify two interference patterns from neighboring links that should be mitigated to perform intelligent channel bonding decisions: co-channel and adjacent channel interference. As a result of our experimentation, we discover the following:

- Due to the MIMO feature in 802.11n devices, performance is not monotonic with RSSI but depends on other factors such as the multipath diversity of the transmission environment. For example, for links with the same signal

strength at varying locations, performance varies significantly, which is in part due to the scattering extent of the environment.

- Packet reception rate (PRR) is a metric that gives clearer insight than RSSI into the quality of a link. For weak links, PRR drops for high transmission rates. However, for strong links, PRR drops only when the scattering environment does not support high transmission rates.

- In an interference-free environment, channel bonding degrades network throughput when the RSSI between a single transmitter and receiver pair is close to the minimum input sensitivity.

- For links in CS range operating on overlapping channels, it is better for stations to compete for the medium with 40MHz-width transmissions to avoid medium-access fairness issues [43] caused by slower 20MHz contenders that occupy the spectrum for longer periods of time.

- For simultaneous transmissions on non-overlapping yet adjacent channels, a 40MHz channel causes more interference from channel leakage than a 20MHz channel.

- For simultaneous transmissions between links with strong signal strengths to each other, a minimum channel separation of 20MHz is necessary to avoid interference.

- For links in CS range operating on overlapping channels with increased load, it is better for stations to compete for the medium with 40MHz-width transmissions, until saturation, at which point performance differences are minimal.

In our evaluation of the performance of channel bonding, we restrict flows in the network to UDP traffic in order to isolate the impact of transport layer parameters on performance and to evaluate the behavior of channel bonding alone. We believe that with the added constraints imposed by TCP, the benefits of channel bonding will be limited to a narrower range of opportunities, given the intolerance of TCP to packet error rates and the susceptibility of a 40MHz channel to interference. We perform a set of studies to develop a better understanding of how channel bonding performs using TCP, and contrary to our expectations, we find that the performance benefits of wider channels apply to both TCP and UDP.

From our experimental study, we identify a metric, called *normalized throughput*, that alerts us to interference patterns in the network. Normalized throughput is the ratio of the achieved throughput over the expected throughput. By monitoring patterns in the behavior of channel bonding in response to specific network

conditions, we observe that normalized throughput is a good indicator of unfavorable network conditions. We therefore use normalized throughput in the design and implementation of an interference detector; this detector can form the foundation to more robust and accurate algorithms that can adapt bandwidth to variations in network conditions. Our proposed detector successfully identifies interference conditions in 100% of test cases.

We evaluate our findings by applying them to a testbed scenario. By understanding the effects of network conditions on the performance of channel bonding, we monitor channel conditions and proactively assess the network to predict when network conditions are favorable to channel bonding and improve performance, as well as when conditions are disadvantageous and degrade performance. We find that, compared with naïve and uninformed solutions, we exploit all possible opportunities for channel bonding and improve network throughput by a factor of more than 80%.

This chapter is organized as follows. We discuss background and related work in Section 5.2. In Section 5.3, we describe the details of our testbed environment and experimentation. We present experimental results in Section 5.4 and discuss observed patterns in channel bonding behavior. Based on our findings, we discuss methods of assessing a network for channel bonding opportunities in Section 5.5. To verify the correctness of our assessment, we provide a proof of concept in Sec-

tion 5.6, where we show that our recommendations for exploiting channel bonding improve network throughput. Finally, we conclude in Section 5.7.

5.2 Background and Related Work

We now present the related body of work. We discuss how channel bonding in the 802.11n standard, unlike in 802.11a/b/g, presents a compelling research direction in the context of wireless LANs. In particular, we focus on how existing work has fallen short in studying the utilization of channel bonding in 802.11n environments.

The 5GHz Frequency Range: Channel bonding in 802.11n combines two adjacent 20MHz channels to form a single 40MHz channel. Ideally, this feature should double the PHY layer data rate. One tradeoff of channel bonding is that fewer channels remain for other devices [15]. In traditional 2.4GHz Wi-Fi deployments where there are only three non-overlapping 20MHz channels, channel bonding has been found to be harmful due to both the limited channel availability and the resulting throughput degradation [99, 103]. There are more opportunities to exploit channel bonding in the 5GHz range where there are 24 non-overlapping 20MHz channels and up to 12 non-overlapping 40MHz channels. Furthermore, unlike the 2.4GHz band which shares its frequency with commonly used consumer products, such as

Bluetooth and microwave ovens, the 5GHz band typically suffers less interference.[1] Our work therefore focuses on operation within the 5GHz band.

MIMO: IEEE 802.11 networks have operated on the 5GHz band since the emergence of the 802.11a standard in 1999. Although 802.11a networks have benefited from the increased number of non-overlapping channels in the 5GHz band, the benefit was not widely realized due to the decrease in transmission range caused by operating at higher frequencies. Furthermore, if an access point (AP) were to take advantage of wider channels to increase the data rate, for example by channel bonding, the AP would consequentially suffer an additional decrease in transmission range as well as greater sensitivity to interference [15].

With the introduction of MIMO smart antenna technology in the 802.11n standard [1], adoption of wider channels is now an appealing concept. Problems that are faced using wider channels in traditional 802.11 SISO networks can be mitigated with MIMO. MIMO utilizes multiple discrete antennas to transmit multiple data streams simultaneously along the same channel [32, 79, 104].[2] MIMO takes advantage of this multiplicity of data streams to improve either the signal-to-noise ratio (SNR) or the data rate at the same distance by using one of its two modes of operation: *spatial diversity* and *spatial multiplexing*, respectively. *Spatial diver-*

[1] WLANs vacate the 5GHz band in the presence of weather and military radar signals. This is called Dynamic Frequency Selection (DFS).

[2] The IEEE 802.11n specification allows up to four spatial data streams; current products in the market support up to three streams.

sity transmits the same signal over multiple antennas simultaneously, while *spatial multiplexing* transmits different signals over multiple antennas.

Previous work has looked at the impact of MIMO on 802.11n testbed environments [39, 84]. Compared to traditional SISO systems, MIMO is shown to improve the transmission range, reliability, and data rate. Some work has focused on the impact of MIMO on the design of rate adaptation solutions [58, 83]. These studies show that traditional methods of determining the best operating rate in a SISO environment no longer apply with MIMO.

Although existing research has uncovered the unique behavior of MIMO systems in 802.11n environments, we have yet to understand the implications of these findings on the performance of channel bonding in 802.11n WLAN settings. We build on these findings to accurately assess the performance of channel bonding in 802.11n WLANs.

Channel Management: The ability of channel bonding to increase data rate can be leveraged to allow more flexibility in distributing the load. This flexibility has defined the recent direction in bandwidth management solutions that advocate adapting channel width in wireless networks to accommodate changes in load conditions [15, 36, 73, 89]. These studies rely on the assumption that increasing the channel width should theoretically increase the data rate, since more data is being transmitted over a wider bandwidth. Recent studies, however, have shown that the

benefits of channel bonding in 802.11n are influenced by network factors, such as interference and loss [7, 85, 99]. Therefore, it is clear that channel management solutions in 802.11n WLANs must first understand the behavior of channel bonding in order to make intelligent decisions as to how to assign bandwidth in the network.

Experimental Studies of 802.11n: Experimental studies on 802.11n have provided valuable insights into its features [85, 99]. Work most related to ours proposes a framework to incorporate channel bonding in WLANs [7]. Yet, although much has been contributed, research still falls short on accurately analyzing and characterizing the behavior of and opportunities for channel bonding in real-world WLAN settings, where interfering links co-exist. Most prior work has evaluated operation on the busy 2.4GHz range [85, 99, 108], which has a limited number of non-overlapping channels; performance constraints are thus tighter to be able to properly gauge performance benefits [15, 99, 103]. As such, a complete picture that demonstrates the opportunities for channel bonding and the effect of varying network conditions on performance has not yet been achieved in wireless networks.

5.3 Test Environment

We set out to understand the characteristics of channel bonding in 802.11n WLANs and the network factors that influence its behavior to ultimately predict how to max-

imize performance. To achieve this goal, we set up a configurable testbed that gives us the flexibility to evaluate channel bonding in a variety of network conditions. Below, we describe our testbed environment while focusing on node configuration, measurement tools and the general measurement setup. Configurations that are specific to particular experimental scenarios are discussed when the findings of those experiments are presented.

5.3.1 Node Configuration

We conduct our experiments using a stationary testbed deployed in a semi-open office environment. The testbed consists of 12 laptops. All the laptops are equipped with an 802.11n AirMagnet 2×3 MIMO PC card with a dual-band Atheros AR5416/AR5133 chipset. The AR5416 baseband and MAC processor supports modulation and coding scheme (MCS) indices 0 to 15 (see Table 5.1 for a detailed list of supported PHY modes). The Linux device driver is based on the Atheros ath9k that supports 802.11n [112].

We vary the locations of transmitter and receiver pairs to obtain a rich set of link conditions, where the transmitter operates in AP mode. Our experiments consist of 20 different links. We set the symbol guard interval to the short guard interval (SGI)

Table 5.1: Tested Modulation and Coding Schemes (MCS).

MCS index	Spatial Streams	Modulation	Coding Rate	Data Rate (Mb/s)	
				20 MHz	40 MHz
0	1	BPSK	1/2	6.5	15.0
1		QPSK	1/2	13.0	30.0
2			3/4	19.5	45.0
3		16QAM	1/2	26.0	60.0
4			3/4	39.0	90.0
5		64QAM	2/3	52.0	120.0
6			3/4	58.5	135.0
7			5/6	65.0	150.0
8	2	BPSK	1/2	13.0	30.0
9		QPSK	1/2	26.0	60.0
10			3/4	38.0	90.0
11		16QAM	1/2	52.0	120.0
12			3/4	78.0	180.0
13		64QAM	2/3	104.0	240.0
14			3/4	117.0	270.0
15			5/6	130.0	300.0

of $400ns$.[3] Our goal in configuring the network is to select link settings that yield the highest PHY data rates supported by the 802.11n standard.

5.3.2 Measurement Environment

In our experiments, we generate constant bit-rate UDP traffic between the transmitter and receiver pairs using the *iperf* tool, with fixed packet sizes of 1500 bytes. We monitor UDP flows, and evaluate their performance in terms of MAC layer throughput and packet reception rate (*PRR*). All our reported performance metrics are averaged over 10 runs. We restrict flows to UDP in order to measure the performance gains of channel bonding without having to account for the performance effects of transport layer parameters, such as TCP's congestion control. Further-

[3]The chipset does not allow SGI to be used with 20MHz channels.

more, to provide accurate measurements of the packet delivery rate at the MAC layer, we disable both link layer retransmissions and frame aggregation (A-MPDU). By disabling aggregation, we also avoid software-driven retransmissions. This system setup constrains the maximum throughput to less than 45Mb/s, even for MCS 15.[4]

We run our experiments for all supported MCS (see Table 5.1) and identify the best MCS for each tested link and channel width configuration. In so doing, we mimic the behavior of an ideal rate adaptation mechanism that selects the MCS that maximizes link performance. We henceforth use the term *best throughput* to reflect the highest application layer throughput yielded by the best MCS for the link under study. We thus present a fair evaluation of the performance of 40MHz versus 20MHz channels under varying network scenarios. We categorize MCS indices into two groups based on their corresponding MIMO mode and refer to these groups as *sets*: a set for MCS 0 to 7, which exploits *spatial diversity*, and a set for MCS 8 to 15, which achieves *spatial multiplexing*.

We conduct experiments exclusively on the 5GHz band and at night when potential for interfering traffic is minimal.

[4]Compliance to the 802.11 standard imposes an irreducible MAC overhead, independent of bandwidth, on every transmitted packet; even with an infinite PHY rate, the maximum throughput will be bound to 50Mb/s. With aggregation, the fixed overhead is shared by multiple frames, reducing the relative overhead, thus allowing higher throughput.

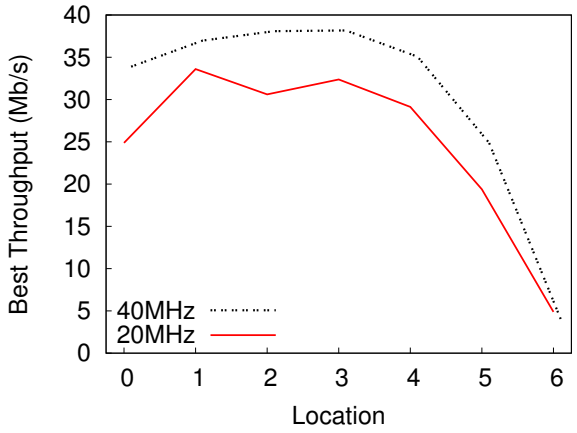


Figure 5.1: Throughput achieved between single transmitter and receiver pairs at varying locations. The locations are sorted in order of decreasing RSSI.

5.4 Empirical Study of Channel Bonding

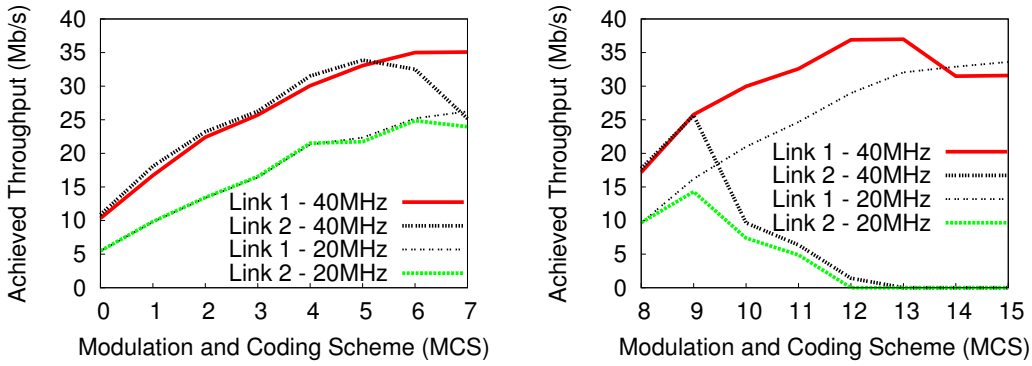
The purpose of our study is to examine the performance of an IEEE 802.11n WLAN with channel bonding in response to particular network characteristics. Our findings give us guidance into how to build 802.11n networks that maximize the performance gains available from channel bonding. In the following subsections, we use experimentation to answer questions that are critical to understand the use of 40MHz channels in 802.11n WLAN environments.

5.4.1 What parameters affect the performance of channel bonding between a transmitter and receiver pair?

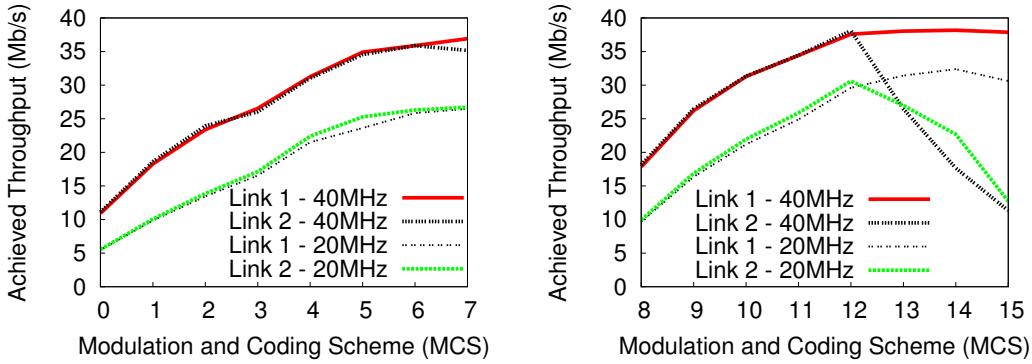
In this section, we take a close look at the parameters between a transmitter and receiver pair that affect the performance of channel bonding.

Is performance always monotonic with RSSI?

Ideally, we expect performance to decrease monotonically as the received signal strength indicator (RSSI) decreases. However, we find that RSSI does not accurately reflect performance, as shown in Fig. 5.1. Fig. 5.1 plots the best throughput between single transmitter and receiver pairs at varying locations, sorted in decreasing order of RSSI of each node pair, from strongest to weakest. Regardless of channel width, locations 1 to 4 in Fig. 5.1 outperform location 0, even though the latter receives the strongest signal. This fact is also observed in Figs. 5.3(a) and (b), which show the PRR and throughput of a link with strong (above -40dBm) and moderate (above -50dBm) RSSI, respectively; the link with moderate RSSI outperforms that with strong RSSI. We can thus affirm that RSSI alone is not an adequate link quality metric, especially at high data rates, where performance with MIMO is further influenced by propagation characteristics. As further discussed in Section 5.4, MIMO transmissions can take advantage of different propagation phenomena. These phenomena depend on particular characteristics of the path between a transmitter and receiver, which can be highly unpredictable.



(a) Good signal quality (> -30dBm)



(b) Moderate signal quality (between -43 and -46dBm)

Figure 5.2: Throughput achieved between the transmitter and receiver pairs with similar signal qualities.

Although RSSI does not directly reflect performance, we find that it is necessary, but not sufficient, information to determine when a 40MHz channel outperforms a 20MHz channel. For RSSI values that are close to the current MCS's sensitivity (which is higher for faster modulations), channel bonding degrades performance. In Fig. 5.1, we observe that only for location 6, which has an average RSSI[5] of -82dBm, a 20MHz channel yields a higher throughput. Since the minimum receiver sensitivity of a 40MHz channel is -79dBm while that of a 20MHz channel is -82dBm, operating on a 40MHz channel at location 6 degrades performance because RSSI falls below the sensitivity range of a 40MHz channel. When the RSSI lies above the minimum sensitivity, channel bonding always improves performance. However, with low RSSI values, the sacrifice in available spectrum to channel bond may not be worthwhile, given the low level of improvement. Section 5.4.2 gives more insight into this matter.

How does rich scattering affect performance?

As shown in Section 5.4, RSSI alone is not a good predictor of 802.11n performance. In this section, we demonstrate how rich scattering contributes to this behavior.

[5]The average RSSI is the per-packet RSSI averaged over multiple received beacon packets, where per-packet RSSI is the RSSI averaged over all MIMO antennas.

Multi-path diversity has traditionally had a negative impact on performance. However, with the incorporation of MIMO technology in 802.11n networks, multi-path diversity is now used to overcome fading effects and instead improve signal quality [32]. We evaluate the impact of MIMO by comparing the throughput achieved between links with similar signal quality. In Fig. 5.2(a), we compare two links with good signal quality (> -30dBm), where the client for Link 2 is in direct line-of-sight of the transmitter while the client of Link 1 is separated by obstacles. In Fig. 5.2(b), we compare two links with moderate signal quality (between -43 and -46dBm), where the receivers are placed at different locations and are separated by different obstacles. The behavior of the links is representative of the behavior observed in our experiments. For the *spatial diversity set* (MCS 0–7), we observe little difference between links of similar strength. However, for the *spatial multiplexing set* (MCS 8–15), we observe considerable differences in throughput. In Fig. 5.2(a), Link 1 and Link 2 achieve similar throughput values for low MCS indices, but for MCS greater than 8, Link 2's performance drops while Link 1 maintains or improves its performance with higher MCSs.

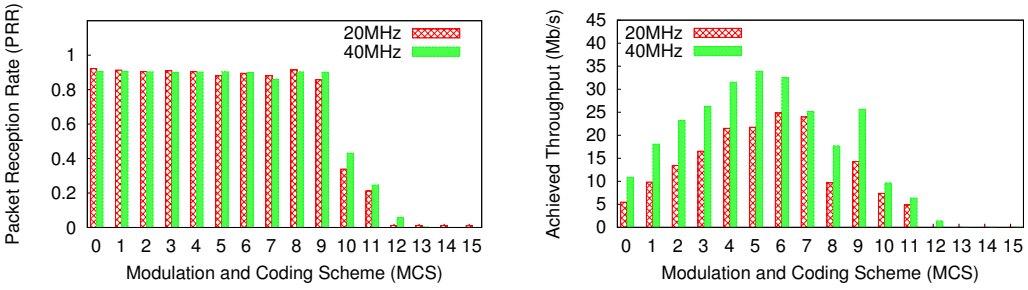
As mentioned in Section 5.2, *spatial multiplexing* transmits multiple independent data streams over different transmit antennas on the same channel. In order for the signals to be correctly decoded, they should arrive at the receiver across independent spatial paths with sufficiently different spatial signatures [79]. Although

there is no existing method that can accurately characterize multipath diversity, we attribute the performance differences in Fig. 5.2 to the extent to which an environment is rich in scattering. The impact of poor scattering is observed more accurately for strong links where the transmitter and receiver are likely to be in close range with each other, as seen in Link 2 in Fig. 5.2(a), where both nodes are in line-of-sight. In such cases, performance varies considerably due to the potential scarcity of independent spatial paths between transmitter and receiver pairs. Yet, regardless of the scattering environment, a 40MHz channel consistently outperforms a 20MHz channel, provided that the link's RSSI is above minimum sensitivity, and the link is configured to its best MCS.

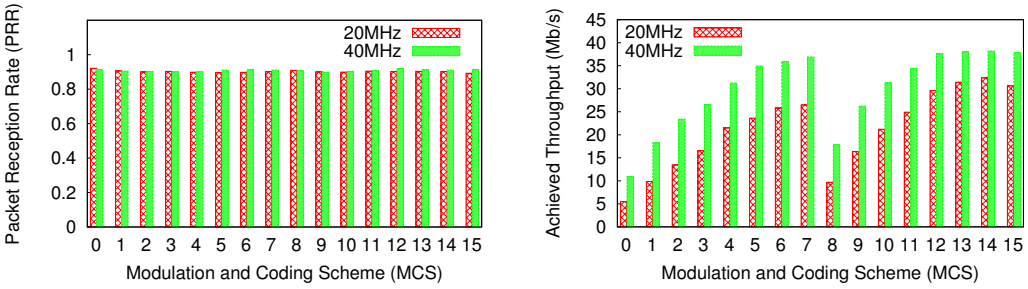
What patterns do we observe between varying MCS values?

We now evaluate our performance metrics, namely PRR and achieved throughput, for all possible MCS values in a variety of link qualities, as shown in Fig. 5.3. The results of our experimentation expose distinct patterns in the behavior of our performance metrics with respect to different MCSs.

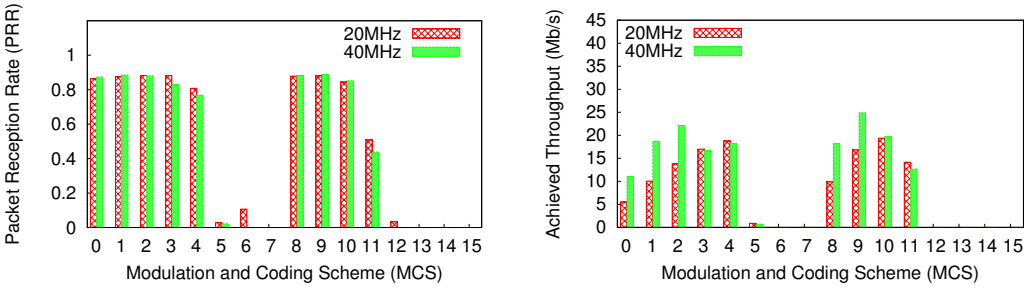
As expected, independent of the signal strength, throughput either monotonically increases or decreases as we move from low to higher transmission rates within each MCS set. Recall that MCS values are divided into two sets based on the MIMO mode



(a) Good signal quality (-30dBm)



(b) Moderate signal quality (-45dBm)



(c) Poor signal quality (-75dBm)

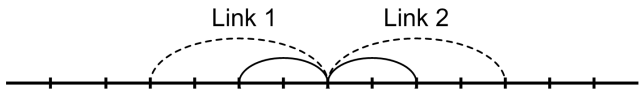
Figure 5.3: PRR and throughput between transmitter and receiver pairs with good, moderate, and low signal qualities.

used (MCS 0 to 7 and 8 to 15). In other words, when throughput begins to decrease at a particular MCS, any higher MCS *in that set* will not perform better.

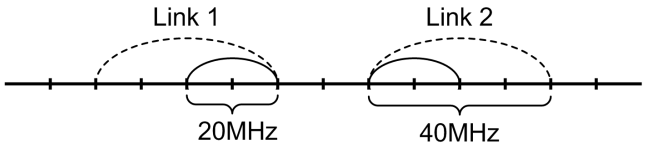
PRR gives clearer insight into the quality of a link than RSSI or throughput. PRR remains relatively constant and then drops when conditions cannot support the required transmission rate at a particular MIMO mode; this behavior is consistent among all links. Fig. 5.3(c) depicts how weak links perform poorly at high transmission rates, irrespective of the MCS set. On the other hand, for strong links that suffer from scarcity of multipath diversity, PRR drops at MCS values that sacrifice data redundancy for higher rates using *spatial multiplexing*, as shown in the PRR plot of Fig. 5.3(a) for MCS above 9. In general, by comparing the behavior of a 40MHz versus a 20MHz channel in Fig. 5.3, it is clear that channel bonding outperforms a 20MHz channel, particularly when the correct MCS is chosen. Doubling the physical rate compensates for the increased error rate provided that, roughly, $\text{PRR}_{20MHz} < 2\text{PRR}_{40MHz}$.

5.4.2 How should bandwidth be assigned between neighboring nodes?

Our evaluation of the behavior of channel bonding in isolation revealed that it improved performance provided that signal quality is greater than receiver sensitiv-



(a) Adjacent transmission channels



(b) Transmission channels separated by 20MHz

Figure 5.4: Separation cases between non-overlapping channels: (a) adjacent channels, and (b) 20MHz channel width apart.

ity. We now evaluate how channel bonding behaves in more realistic settings with neighboring and potentially interfering links.

The impact of neighboring links depends on the amount of spectral overlap. This phenomenon has been studied extensively, particularly in the 2.4GHz band, in the context of partially overlapping channels [72]. We now evaluate how neighboring links with varying bandwidth impact performance, in order to be able to assign channels efficiently. To do so, we examine two constituent subproblems: how to

Table 5.2: Effects of channel leakage on performance.

Row	Link Conditions	Link Metrics	Bandwidth of Studied Link									
			20MHz					40MHz				
			inter-ference-free	20 MHz adj	20 MHz sep	40 MHz adj	40 MHz sep	inter-ference-free	20 MHz adj	20 MHz sep	40 MHz adj	40 MHz sep
1	Strong Signal Quality and Strong Interferer	Mb/s*	33.60	23.86	30.70	14.18	22.35	36.98	18.74	35.98	20.56	32.69
		MCS	15	13	13	14	6	13	9	6	13	5
		PRR	0.92	0.90	0.90	0.88	0.90	0.91	0.90	0.91	0.91	0.91
2	Moderate Signal Quality and Weak Interferer	Mb/s*	30.61	20.49	26.02	25.12	25.99	38.09	37.42	38.11	37.68	38.44
		MCS	12	4	12	12	12	12	12	12	12	12
		PRR	0.93	0.96	0.95	0.94	0.95	0.95	0.96	0.95	0.96	0.95
3	Moderate Signal Quality and Strong Interferer	Mb/s*	32.27	24.49	29.57	15.08	25.33	38.17	25.21	34.83	29.62	34.00
		MCS	14	15	15	4	12	14	7	12	5	12
		PRR	0.90	0.86	0.88	0.91	0.86	0.91	0.86	0.91	0.92	0.90
4	Weak Signal Quality and Strong Interferer	Mb/s*	19.38	11.34	16.45	12.31	15.51	24.89	11.91	21.58	15.35	23.35
		MCS	10	9	3	3	3	9	2	9	9	9
		PRR	0.84	0.90	0.90	0.90	0.90	0.89	0.90	0.82	0.90	0.89

*: Best Throughput (Mb/s)

assign non-overlapping channels between neighboring nodes, and how to deal with co-channel interference.

What is the impact of channel leakage?

To maximize throughput, simultaneously transmitting neighboring nodes should operate on non-overlapping channels in order to avoid contention and interference for the wireless medium. However, nodes that operate on non-overlapping, yet adjacent, channels, as depicted in Fig. 5.4(a), still suffer interference from channel leakage when power from transmissions on adjacent channels spills to neighboring channels [57].

In Table 5.2, we evaluate the impact of channel leakage on the performance of links with strong, moderate, and poor signal quality. We test channel leakage under conditions where the interferer has both a strong and weak signal quality to the transmitter and receiver of the studied link, as well as when the interferer is operating on either a 20MHz or 40MHz channel. We vary the separation between the non-overlapping channels from being adjacent (*adj*), shown in Fig. 5.4(a), to being separated by a 20MHz channel (*sep*), as in Fig. 5.4(b). We also include the case where the transmission channels are far enough apart (40MHz or more) to be considered interference-free. Table 5.2 shows how these conditions affect the studied link's best throughput, its corresponding best MCS and PRR. These performance

values summarize the methodology we use to conclude patterns in the behavior of non-overlapping channels.

Even in the presence of a weak interferer, performance is still negatively impacted, as shown in Table 5.2, row 2 for a 20MHz link. As the interferer's signal strength increases[6], the performance of the studied link further deteriorates, even when channels are non-adjacent, as shown in Table 5.2 row 1, 3, and 4 for strong interferers. To achieve interference-free conditions, links with strong to moderate signal strength should thus be separated by at least 40MHz.

Typically, power leakage from neighboring transmissions produces reception errors due to the decreased SINR (Signal to Interference-plus-Noise Ratio). The increased error rate can be compensated by using a more reliable (but slower) modulation. Furthermore, when interfering transmissions on adjacent channels are from physically close nodes, power leakage could be strong enough to activate carrier sensing at the transmitter's MAC layer [10, 57]. By activating carrier sensing, collisions are avoided, and the transmitter can use more aggressive modulations, which compensates for the negative impact of deferred transmissions. As mentioned earlier, for the same interferer, a 20MHz transmission has more energy than a 40MHz transmission and, thus, a 20MHz transmission is more easily detected. Therefore, for sufficiently strong interferers that activate carrier sensing, performance is better

[6]The RSSI of the interferer link is measured at the studied link from beacon packets, which are sent at a constant bit rate on a 20MHz channel.

with a 20MHz interferer than with a 40MHz interferer, as shown in Table 5.2 row 1, 3, and 4[7]. However, if the studied link channel bonds, its best MCS is generally less aggressive and thus more robust to interference. In such cases, collisions will not significantly impact performance and *40MHz adj* performs better than *20MHz adj*. On the other hand, if the interferer is weak, as shown in Table 5.2 row 2, and the power leakage is seldom above the carrier sensing threshold, a 40MHz interferer produces fewer reception errors since it is received with less energy.

Table 5.2 demonstrates that channel bonding must be intelligently executed to improve performance. In some cases, even if a free 40MHz channel is available, leakage from adjacent channels can degrade performance compared to that of a single 20MHz channel. For example, in Table 5.2 row 1, although the studied link is strong, if the interferer is strong and operates on an adjacent 20MHz channel (*20MHz adj*), then channel bonding degrades performance. On the other hand, if the interferer operates on an adjacent 40MHz channel (*40MHz adj*), channel bonding improves performance. This observation applies independent of the signal strength of the interferer, as shown in all cases in Table 5.2. Further, if the interfering channel is separated by 20MHz, channel bonding always improves performance.

[7]In Table 5.2 row 4, there is little difference between *40MHz adj* and *20MHz adj* for a 20MHz channel, since the studied link operates using low, reliable MCS. This link is thus more resilient to interference caused by a lower-energy *40MHz adj* leakage.

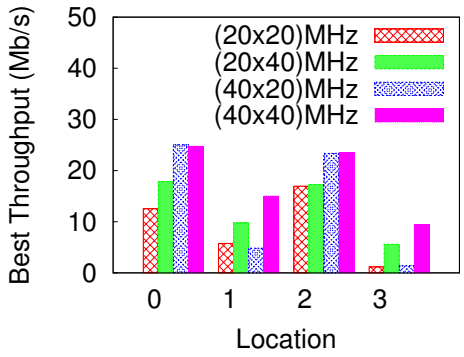


Figure 5.5: Best throughput for different links suffering from co-channel interference. The legend is defined as *(transmitter's bandwidth* $\times$ *interferer bandwidth)MHz*. The locations are sorted in order of decreasing RSSI.

What are the effects of sharing the channel?

In densely populated networks, devices share channels, since the number of available non-overlapping channels may not be enough to avoid co-channel interference. Cells may in fact share a channel without being aware due to the known hidden terminal problem, which is a difficult to detect without client-side modifications [71]. We now investigate the hidden terminal problem that occurs when transmitters are not in transmission range of each other, but in carrier sensing range. We evaluate the impact of channel bonding on the performance of such shared channels.

We configure the network such that two transmitters share the wireless medium. We vary the channel width of each transmitter and evaluate the throughput when the channels completely overlap in Fig. 5.5. The 802.11n specification for channel bonding pairs in the 5GHz range states that 40MHz transmissions cannot partially

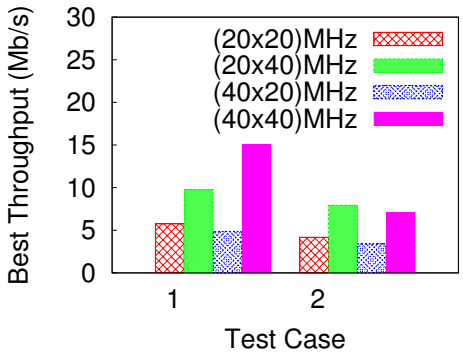


Figure 5.6: Best throughput for a link (location 1 in Fig. 5.5) suffering from co-channel interference. In Test Case 1, the overlapping transmitter has good link quality to its receiver and operates at MCS 10. In Test Case 2, the overlapping transmitter has poor link quality to its receiver, and thus operates at MCS 0.

overlap with each other [1]. For simplicity, we refer to the transmitter under question as *T* and the transmitter sharing the channel with *T* as *S*. We define the legend in Fig. 5.5 as: *(T channel-width* $\times$ *S channel-width)MHz*. We vary the signal strength between *T* and its corresponding receiver and order the locations by decreasing signal quality. *S* always has good signal quality to its receiver and operates at high transmission rates.

The best performance in Fig. 5.5 occurs when both *T* and *S* operate on a 40MHz channel (*40*$\times$*40 MHz*). In most cases, *T*'s operation on a 40MHz channel, independent of the bandwidth of *S*, improves performance compared with a 20MHz channel; however, this condition is not guaranteed and depends on how effectively a link can take advantage of signal strength to increase data rate, as discussed in Section 5.4.1. For example in Location 1, *T*'s performance degrades with channel bonding when

it competes for the medium with a 20MHz interferer (*40×20 MHz*). In this case, performance degrades due to the combined effects of interference and channel sharing, resulting from *S* being a weak interferer. When sharing a channel with a weak interferer, not all transmissions can be detected, and thus the "effective" noise on the shared channel will increase; the increased errors in 40MHz forces *T* to use slower MCS.

In situations where multi-rate CSMA nodes share the medium, since all transmitters have the same access rights, low data rate nodes have been found to capture the medium for longer periods of time, thus penalizing fast stations [43]. Therefore, we also evaluate the scenario where, instead of operating at high data rates, *S* operates at low data rates, shown in Fig. 5.6. Test cases 1 and 2 correspond to the scenario in location 1 of Fig. 5.5; however, in Test case 2, *S* now operates at the lowest data rate of MCS 0. As we can see, when *S* operates at low rates, *T* does not improve performance by channel bonding.

Our findings on channel sharing show that, regardless of the bandwidth of *T*, it is more advantageous for *T* to compete for the channel with an interferer who transmits at 40MHz: 40MHz interferers attain higher transmission rates and alleviate fairness issues in multi-rate scenarios, leading to better performance. However, the decision to channel bond relies on the accurate characterization of *T*'s potential to take ad-

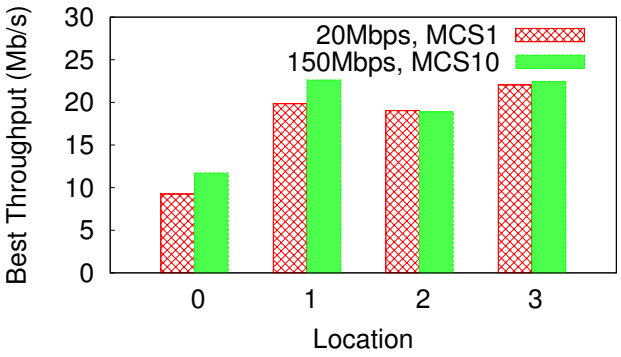


Figure 5.7: Best throughput for different links suffering from a 40MHz co-channel interferer and fairness constraints. The locations are sorted in order of decreasing RSSI.

vantage of channel bonding, as described in Section 5.4.1, as well as knowledge of the transmission rate of S with its corresponding receiver.

5.4.3 How does channel utilization affect performance?

In channel sharing conditions, a station's medium access opportunities depend on the load imposed on the network by other interferers operating on the same or overlapping channels. We now evaluate the impact of different load levels on the performance of channel bonding. We configure the network with two transmitters sharing the wireless medium, and use a configuration identical to the one in Section 5.4. We again refer to the studied transmitter as T and the transmitter sharing the channel with T as S.

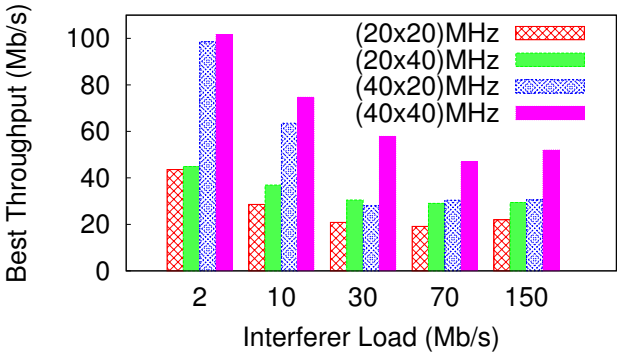


Figure 5.8: Best throughput for a link suffering from co-channel interference with varying interferer load. We define the legend: *(transmitter's bandwidth* $\times$ *interferer bandwidth)MHz.*

In order to isolate the effect of channel utilization on performance, we enable packet aggregation in these experiments so that transmitters with a high MCS are allowed a higher degree of aggregation. This means that fast STAs can send more aggregated frames per transmission opportunity than slow STAs, enabling airtime fairness [**?**]. As we show in Fig. 5.7, packet aggregation mitigates the impact of fairness issues caused in such multi-rate scenarios and allows us to evaluate the impact of channel utilization alone. Under the same channel utilization conditions for the interferer *S*, *S* at MCS 1 impacts the performance of *T* similarly to when *S* operates at MCS 10.

We make two key observations as we evaluate the impact of load on performance in Fig. 5.8. Our first observation is that the benefit of channel bonding decreases with increased load. For the same interferer bandwidth and rate, the benefit of channel

bonding decreases as the load imposed by the interferer increases and the link approaches saturation. In saturation, there is high contention for the shared medium and little time to exploit the benefits of channel bonding.

Our second observation is that, with increased load, it is still better for stations to compete for the medium with a 40MHz interferer, until saturation, at which point performance differences are minimal. As discussed in Section 5.4, competing with a 40MHz interferer reduces fairness issues due to higher PHY rates achieved by channel bonding. The highest throughput is achieved when both T and S operate on completely overlapping 40MHz channels (*40x40 MHz*).

Our findings reveal that our channel bonding decisions will differ depending on the channel utilization of the overlapping channel. For example, a transmitter might choose to compete with a 20MHz interferer with low load instead of a 40MHz interferer with higher load.

5.4.4 What are the performance benefits of channel bonding using TCP traffic?

We now evaluate the performance of channel bonding under TCP traffic. TCP is more sensitive to packet losses, and we therefore evaluate the impact of varying PRR levels on performance. In Table 5.3, we show a representative sample of our performance measurements from 15 different links using TCP traffic. We compute

Table 5.3: Comparison of TCP and UDP performance.

Link Metric	Link Strength	UDP		TCP	
		20 MHz	40 MHz	20 MHz	40 MHz
Achieved Throughput (Mbps)	Strong	29.66	32.35	21.72	24.26
	Moderate	22.94	27.46	17.44	21.22
	Weak	16.50	25.66	12.00	20.22
Best MCS	Strong	15	13	13	13
	Moderate	12	12	12	11
	Weak	5	11	10	11
Throughput Gain (%)	Strong	9		12	
	Moderate	20		22	
	Weak	56		69	

the achieved throughput of each link, the corresponding MCS that achieves that throughput value, which we refer to as the *Best MCS*, as well as the channel bonding throughput gain. We also include UDP results for the same links to compare against.

As expected, we find that for varying link strengths, and for the same transmission bandwidth, TCP throughput values are lower than UDP throughput values. This result is due to TCP's increased overhead and higher sensitivity to packet losses. However, if we look at the throughput gains from channel bonding, we find that the performance benefits under UDP traffic are also observed under TCP traffic. Though a 40MHz channel is more susceptible to loss [7], channel bonding is capable of achieving higher throughput values for all link strengths. Furthermore, we observe that the performance improvements achieved by UDP over TCP are similar for both 20MHz and 40MHz channels, provided that a more conservative MCS is used to counteract both the increased errors caused by channel bonding as well as

TCP's sensitivity to errors. As such, channel bonding does not appear to particularly affect TCP.

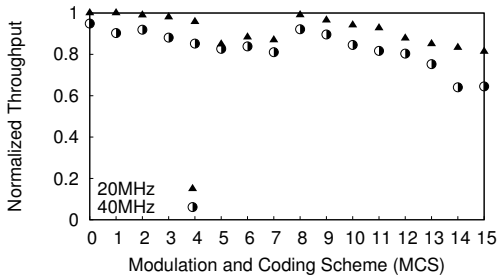
Along the same lines, we also observe that, regardless of the traffic type, the MCS chosen by a 20MHz channel is generally more aggressive than that chosen by a 40MHz channel[8]. Since a 40MHz channel is more susceptible to interference, it utilizes more reliable (*i.e.*, low) MCS rates in order to maximize performance.

Though a 40MHz channel is more susceptible to noise and interference, and TCP traffic suffers from greater sensitivity to loss, the combination of TCP and channel bonding still provides performance benefits as compared to a 20MHz channel. This result demonstrates that the performance improvements of channel bonding are not restricted to a particular traffic type. Hence, 40MHz channels can be exploited in WLANs for both UDP and TCP traffic in order to achieve higher data rates.

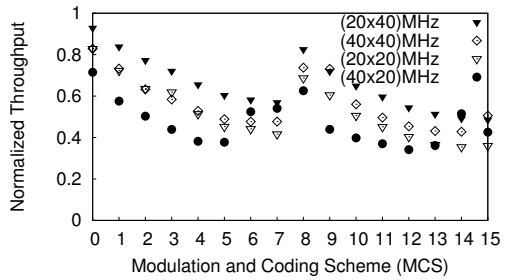
5.5 Identifying Channel Bonding Opportunities

Through our investigations in Section 5.4, we identified the network characteristics that are either conducive or detrimental to the performance of channel bonding. With this knowledge, we now answer some questions that allow us to evaluate a network to determine channel bonding opportunities and to make recommendations of

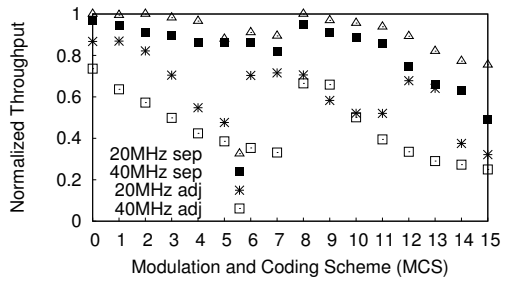
[8]MCS 8–15 apply the same modulation and coding as MCS 0–7.



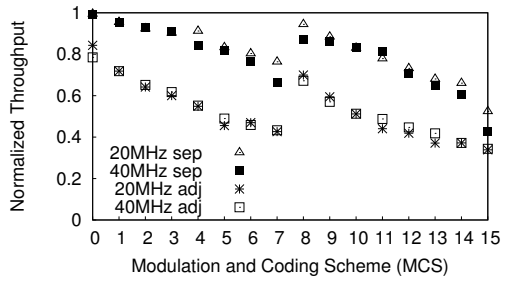
(a) Interference-free environment



(b) Channel overlap with another transmission



(c) Channel leakage from a neighboring 20MHz transmission



(d) Channel leakage from a neighboring 40MHz transmission

Figure 5.9: Normalized throughput of a moderate strength link.

when channel bonding improves the performance. This information could be used as valuable input to a channel management scheme.

5.5.1 How can unfavorable network conditions be determined from performance metrics?

There are multiple conditions in WLANs that contribute to performance variations. Of these conditions, some can be mitigated through intelligent channel management solutions without readjustments to the network topology nor client-side modifications; we refer to these conditions as *unfavorable network conditions*. In our work, we identify two possible unfavorable conditions. One condition is the presence of nodes that operate on overlapping channels. The second condition is interference caused by channel leakage from nodes operating on adjacent channels. As shown in previous sections, both conditions lead to degradations in performance if left unidentified and unresolved.

In the evaluation of our results, we define *normalized throughput*, an accurate indicator for unfavorable network conditions. Based on *normalized throughput*, we design and implement a MAC-layer anomaly detector that successfully alerts to the presence of unfavorable network conditions in 100% of the test cases. This detector can form the foundations of future channel management algorithms.

Normalized Throughput

Normalized throughput is the ratio of the achieved throughput over the expected throughput. Expected throughput is the throughput that would be achieved in an ideal environment. We measure achieved throughput at the MAC layer. Similar to [31], we calculate expected throughput (E_{Th}) in terms of delay per packet:

$$E_{Th} = \frac{K \cdot L_{data} \cdot PRR}{DIFS + T_{BO}(PRR) + T_{Kdata} + SIFS + T_{ACK}} \tag{5.1}$$

where K is the number of aggregated frames, which is equal to 1 with disabled aggregation; L_{data} is the payload carried per frame (in bits); $DIFS$ is the time interval a wireless medium should be idle before a station can transmit; T_{BO} is the average backoff time, which is a function of the PRR; T_{Kdata} is the total time required to send the A-MPDU (including preamble and headers) at a given PHY rate; $SIFS$ is the constant time interval between a data frame and its ACK; and T_{ACK} is the time required to send an ACK frame (or Block ACK).

We observe from our results that normalized throughput is a good indicator of unfavorable network conditions; the greater the impact an unfavorable condition has on performance, the more clearly the impact is reflected in the computed normalized throughput at each MCS.

Fig. 5.9 depicts the typical behavior of normalized throughput for all MCS under varying network conditions. Fig. 5.9(a) computes normalized throughput for a single link in an interference-free environment. For that link, Fig. 5.9(b) represents values when a second link operates on an overlapping channel, while Figs. 5.9(c) and (d) show values when a second link operates on a non-overlapping, yet adjacent, channel. We find that the behavior of a link in an interference-free environment is consistent, independent of the strength and conditions of that link. This observation allows us to identify and characterize situations where performance is affected by unfavorable network conditions. We now explain our observation and reasoning.

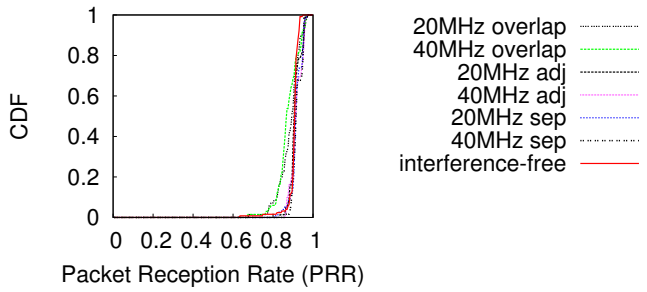
Fig. 5.9(a) depicts the behavior we observe in an interference-free environment. We note a gradual drop in normalized throughput as transmission rates increase. For low MCS, particularly for MCS values of $0, 1, 2$ and $8, 9, 10$, the achieved throughput very closely approximates the expected throughput with ratios between 90% and 100%. This condition holds as long as the RSSI of the link in question is greater than the receiver's minimum input sensitivity. However, as rates increase, ratios monotonically drop. Furthermore, we observe that 20MHz channels achieve higher ratios than 40MHz channels for all MCS. Therefore, we believe that the distance of the achieved throughput from the expected throughput is due to the strict SNR requirements necessary to achieve those rates.

The difference between achieved and expected throughput increases depending on the severity of the aforementioned penalty imposed on fast stations due to sharing a medium with slow stations. Therefore, even with a high PRR, the achieved throughput will be lower than expected. If we look at Figs. 5.9(b), (c), and (d), we notice a consistent pattern, which is the drop in normalized throughput for low MCS, which we do not observe in interference-free settings. This drop is reflected in the higher transmission rates where normalized throughput drops more steeply.

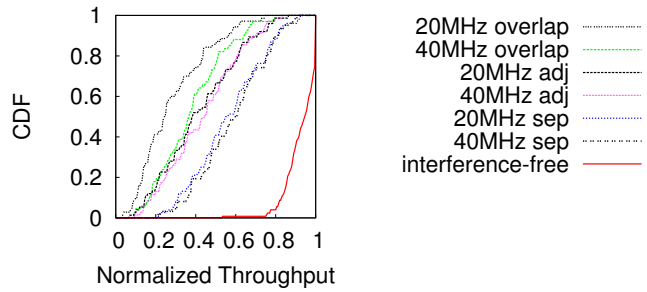
Next, we test the effectiveness of using normalized throughput in the design of a MAC-layer anomaly detector.

Network Anomaly Detector

We now discuss our network anomaly detector. We start by describing how normalized throughput forms the foundations of our detection mechanism, using results shown in Fig. 5.10. We plot the CDFs of the PRR and the corresponding normalized throughput values for multiple links of varying strength, channel width, and MCS. We further subject our links to the different network conditions investigated in this work, namely channel sharing, leakage, and interference-free conditions. To emulate an intelligent rate adaptation solution that reacts to changes in PRR [21,25,112], Fig. 5.10 only includes the results from MCS values that provide reasonable PRR levels for each link.



(a) CDF of packet reception rate (PRR)



(b) CDF of normalized throughput

Figure 5.10: CDF of PRR and the corresponding normalized throughput values for multiple links of varying strength, channel width, and MCS.

As we can see from Fig. 5.10(a), the PRR achieved by these links under varying network conditions show little difference compared to the interference-free scenario. However, although a successful rate adaptation solution can maintain an acceptable PRR despite changes in network conditions, the corresponding CDF of normalized throughput values for the given PRR values depicts considerable differences in behavior, as shown in Fig. 5.10(b). In interference-free conditions, around 80% of the links have normalized throughput values greater than 0.9, and almost all links have values greater than 0.8. This means that the achieved throughput closely approximates the expected throughput in interference-free conditions. However, in the presence of interference, normalized throughput is distributed over a wider range of values and, in the best case, less than 10% of links attain values greater than 0.8. This increasing difference between achieved and expected throughput reveals the presence of an interferer.

We use this insight to implement a MAC-layer detector that monitors changes in normalized throughput. To compute the expected throughput in time, we used the per-packet transmission and reception statistics from Eq. 5.1. The achieved throughput is computed as the total number of bytes received in a given time, divided by that time. By averaging the achieved throughput over time, we avoid rapid reconfigurations due to non-persistent interfering sources, thus preventing unnecessary

and costly channel migrations. Such highly dynamic interference conditions can be efficiently handled by a fine-grained per-packet rate adaptation mechanism [21].

By subjecting our MAC-layer detector to changing network conditions, we find that it successfully identifies unfavorable network conditions, or anomalies, in the environment in 100% of the test cases. With such high success rates, this detector can form the foundations of future channel management algorithms.

It is worth noting that monitoring changes in normalized throughput to detect anomalies is a useful tool in cases where T is fully saturated. For unsaturated transmitters, the detector could be adapted to consider other metrics, for example medium access delay.

5.5.2 Which parameters characterize a network to determine opportunities for channel bonding?

We compile a list of parameters that facilitate network characterization. This characterization can be applied in both centrally managed and distributed network environments.

Signal strength at receiver (RSSI): Our results show that RSSI is a prerequisite to determining whether 40MHz transmission could improve performance. If RSSI is above the minimum input sensitivity of a 40MHz channel (depends on MCS) in

an ideal environment with minimum interference, a 40MHz channel always outperforms a 20MHz channel.

MCS in use: Since the minimum receiver sensitivity varies according to the MCS in use (higher for faster modulations), a proper selection of the MCS helps to maximize the benefits of channel bonding. In other words, to get the most from channel bonding, it should be set jointly with rate adaptation.

Strength of interfering transmissions: This metric is crucial to determine whether to bond. For example, neighboring links with strong signal strengths to each other will benefit from operating on non-overlapping channels separated by at least 20MHz, to avoid interference from channel leakage.

Physical rates of links in CS range: Beyond the increased contention, links that operate on the same or on overlapping channels, are susceptible to fairness issues in multi-rate scenarios. Knowing the PHY rate of neighboring links is required not only to make good decisions on when to channel bond, but also on which channel should be used.

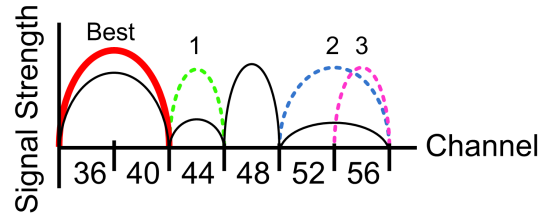
5.5.3 Can performance on a 40MHz channel be inferred from performance on a 20MHz channel?

Due to multipath diversity in wireless environments, transmissions are susceptible to frequency-selective fading. Frequency-selective fading occurs when signals

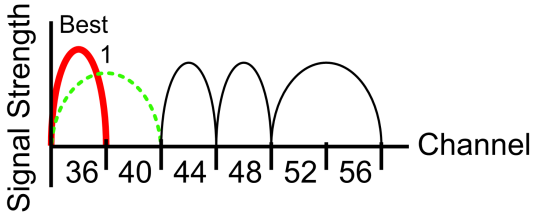
from different paths combine destructively at the receiver and the effect of signal-cancellation is deepest only at particular frequencies. Frequency-selective fading is an unpredictable factor in network environments and degrades performance [40]. Wider channels are thus more susceptible to frequency-selective fading. For the above mentioned reasons, performance from a 20MHz channel cannot be used to infer performance on a 40MHz channel, and we have further confirmed this behavior through experimentation.

5.5.4 Should we increase channel width to 40MHz with incomplete knowledge of the neighboring 20MHz channel?

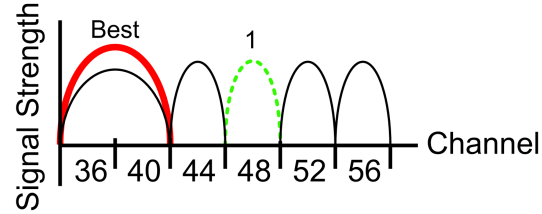
Based on the data presented so far, the answer is clearly *no*. Not only information on the status of the adjacent channel is required due to channel leakage (cf. Section 5.4), but even interfering transmissions on separate channels could potentially affect channel management decisions. If channel bonding is performed under unfavorable conditions, performance will degrade. Particularly, if a 20MHz channel bonds with a channel that is used by a transmitter in carrier sensing range, the medium would then be shared by both transmitters. If the transmitter in carrier sensing range operates at a low physical rate, then performance suffers further due to fairness issues in multi-rate scenarios. As discussed in Section 5.5.2, there are net-



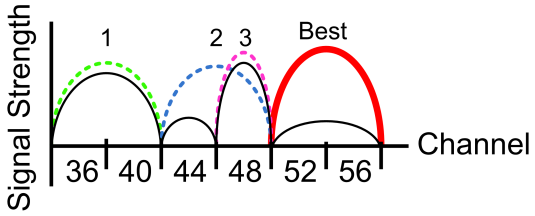
(a) Test Case 1



(b) Test Case 2



(c) Test Case 3



(d) Test Case 4

Figure 5.11: Scenarios to demonstrate the impact of intelligent channel bonding decisions on network performance. In each case, a node T requests bandwidth. The amplitude of signals represents their strength at T. The bold lines represent our suggested channel configurations for T, while the numbered dotted lines indicate possibilities for naïve channel assignments.

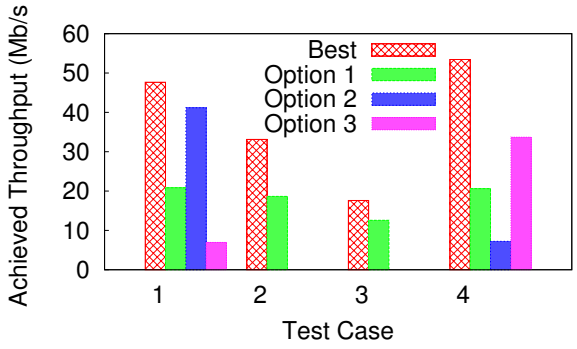


Figure 5.12: Comparison of *T*'s performance using intelligent channel bonding decisions versus naïve approaches.

work parameters that should be identified to perform an intelligent assignment of channel widths to improve network throughput.

5.6 Evaluation of Intelligent Channel Bonding

To demonstrate the impact of intelligent channel bonding decisions on network performance, we create network scenarios where naïve uninformed solutions to channel management lead to incorrect and detrimental decisions. We show that our understanding of channel bonding allows us to make intelligent decisions that leverage the benefits of channel bonding in typical 802.11n environments. We present four different test case scenarios, depicted in Fig. 5.11. In each test case, we characterize the network environment and, accordingly, decide on a channel assignment for a single node *T*. We then evaluate *T*'s performance using our intelligent approach and compare it with *T*'s performance from naïve channel management decisions. It

is worth noting that the same logic we apply for a single node can also be applied in the context of a centrally-managed network. We restrict our analysis to one link since our aim is to demonstrate a proof-of-concept.

For each test case scenario, we depict the corresponding assignment of channels to links in the network, and indicate the possible assignments for T using our intelligent approach (in bold) and a possible set of naïve alternatives (dashed). The strength of the active links with respect to T is represented by the amplitude of the signal. All transmitters are driven to saturation to gauge the capacity of each link. We limit the number of available channels to recreate contention for bandwidth in a large-scale testbed. In all links, RSSI is above the minimum receiver sensitivity (cf. Section 5.4). Furthermore, in these experiments, we enable frame aggregation and automatic rate selection to replicate the behavior of typical off-the-shelf devices. The performance results from each possible channel selection for T, for each test case, are shown in Fig. 5.12.

Case 1, Fig. 5.11(a): All available channels are occupied. To minimize interference, a naïve approach would scan the available channels and assign T the channel on which the weakest interfering signal is received. In this case, T can be assigned a single 20MHz channel at either channels 44 or 56: *Option 1* or *Option 3*, respectively. T could also be assigned bonded channels 52 and 56: *Option 2*. On the other hand, our intelligent solution identifies an opportunity to maximize performance by

channel bonding on channels 36 and 40, where the existing transmitter also operates with a 40MHz channel: *Best*. Intelligent channel bonding will eliminate *Option 2* because the strong adjacent 20MHz transmission at channel 48 will cause interference from channel leakage. *Option 1* is disregarded for the same reason. As for *Option 3*, we do not distinguish any added benefit over *Best*; knowledge of the MCS used by the interfering transmitters would be a key factor for deciding between both options (cf. Section 5.4). As shown in Fig. 5.12, our intelligent solution maximizes performance considerably, with up to 7 factor increase in achieved throughput compared to the naïve solutions.

Case 2, Fig. 5.11(b): Two channels are free. A naïve decision would assign *T* the free 40MHz channel: *Option 1*. However, our study indicates that interference from channel leakage from the neighboring 20MHz transmitter on channel 44, which has a strong signal strength to *T*, can degrade performance. Therefore, our intelligent channel bonding solution assigns channel 36 to *T*: *Best*. As shown in Fig. 5.12, our intelligent solution improves performance by a factor of 83%, from 18Mb/s to 33Mb/s.

Case 3, Fig. 5.11(c): Only one unoccupied 20MHz channel. Similar to Case 2, a naïve approach would assign the free 20MHz channel 48 to *T*: *Option 1*. In this case as well, performance can degrade due to interference from channel leakage from the two neighboring 20MHz transmissions, on channels 44 and 52, with strong signal

strength to *T*. The alternative identified by our intelligent approach is to transmit on a 40MHz-width channel, on channels 36 and 40, in parallel with an existing 40MHz transmission operating at a high physical rate: *Best*. As shown in Fig. 5.12, by identifying the opportunity for channel bonding, we increase the performance by 38%, from 13Mb/s to 18Mb/s.

Case 4, Fig. 5.11(d): We now evaluate the impact of channel utilization. This test case scenario is identical to the one used in Case 1, except we now vary the channel utilization of each interferer. A naïve approach would ignore the impact of channel utilization; thus, its assignment decisions would not differ from those in Test Case 1. In this test case, the interferer on channels 36 and 40 operates at 80% channel utilization, the interferer on channel 44 at 60%, on channel 48 at 50%, and on channel 52 and 56 at 80%.

The decision we made in Case 1, which is operation on channels 36 and 40, no longer achieves the best performance, as shown in Fig. 5.12. *T* now competes for about 20% of the channel with a 40MHz interferer with low MCS, which starves *T*. Similarly, *T* in *Option 3* competes for around 50% of the channel with a 20MHz interferer, which we have shown creates fairness issues. We also evaluate *Option 2*, where *T* operates on a 40MHz channel and contends for the medium with two unsynchronized 20MHz interferes; in such cases, it has been shown that the 40MHz channel will starve [87]. Our intelligent solution identifies an opportunity to max-

imize throughput by competing with the transmitter on channels 52 and 56, which operates at average MCS with high load: *Best*. As shown in Fig. 5.12, we provide up to a 6 fold increase in throughput by also considering the impact of channel utilization on performance.

5.7 Conclusion

Channel bonding in 802.11n networks promises increased data rates and improved performance. In this work, we identify a key set of network factors that allow us to accurately assess the impact of network conditions and channel bonding choices on performance, specifically under 5GHz operation. We find that intelligent channel bonding decisions rely on the knowledge of a transmitter's surroundings, particularly the signal strength of links, interference patterns, and channel utilization. Such findings serve as usage-terms for intelligently incorporating 40MHz operation in network deployments to maximize performance and efficiency. In so doing, we identify the critical principles and rules of a *medium access approach that can exploit 802.11 high speed links, provided through next-generation, 802.11 wider channel widths*.

We further analyze the behavior of channel bonding under TCP traffic loads, and find that the performance values are diminished compared to performance under

UDP. However the benefits of wider bandwidths still hold. Our work serves as a solid foundation on which channel management solutions for 802.11n networks can be built, calling on channel management design principles from existing literature [7, 73], thus enabling further exploitation of high speed 802.11 links. Our findings can be applied both at a network scale to improve channel management of the whole WLAN, and also at a link scale to aid per-packet rate adaptation mechanisms aimed at optimizing individual transmitter and receiver pairs [21]. We believe our work will also apply to the upcoming 802.11ac standard that allows up to 160MHz bonding channels in the 5GHz band.

Chapter 6

Rate Adaptation in 802.11n MIMO Networks

6.1 Introduction

In Chapter 5, we study the performance benefits and usage conditions of wider channel widths in the framework of channel bonding in 802.11n WLANs [20, 21]. We found that the performance of wider channel widths is influenced by the operating rate. This finding motivates the need for a solution that jointly adapts both rate and channel width, in order to enable a system that identifies opportunities to maximize performance. In this chapter, we leverage this insight and our knowledge of 802.11n behavior in the design of an effective joint rate and channel width adaptation solution for 802.11n MIMO WLANs.

A Rate Adaptation (RA) solution selects the best physical bitrate based on time-varying channel qualities. In comparison to RA in legacy 802.11 a/b/g client, the

emergence of the IEEE 802.11n standard has introduced significant increase in sophistication and complexity to WiFi technologies, that require novel approaches to RA. RA in 802.11n networks not only needs to choose the operating rate, but also the channel width and MIMO mode. Using MIMO, a solution can send a single stream using *spatial diversity* to improve signal strength, or multiple simultaneous streams using *spatial multiplexing* to increase the transmission rate.

There are two main approaches to RA: open-loop and closed-loop. In open-loop RA, the transmitter estimates the best rate of the link to the receiver by building on some set of parameters or metrics measured at the transmitter [113]. A closed-loop RA is one in which the receiver's insight into the channel conditions contributes to determining the rate.

As networks become more complex, the use of open-loop RA techniques becomes increasingly inaccurate. An RA solution now has to account for many variables that a transmitter alone cannot accurately capture. In legacy clients, RA mechanisms have to choose among four PHY rates in 802.11b and eight rates in 802.11a/g, whereas 802.11n allows at least 64 combinations from 32 rates and 2 channel widths. By allowing the receiver to contribute to the RA process, we gain an accurate understanding of environment conditions, and the transmitter can more efficiently select the appropriate rate for the link [40].

Perhaps the best RA solution for MIMO environments is to use 802.11n's Channel State Information (CSI) feedback from the receiver to compute the transmission rate [40]. However, complete CSI information is costly to obtain and store [18] and is therefore supported by very few 802.11n devices. Existing RA solutions adopt a practical approach and use a credit-based system [76] or rate sampling [25, 83, 112]. Instead of adapting the rate based on understanding the impact of environment conditions on 802.11n features, these solutions rely on certain heuristics to converge to the best rate, which can be costly or misdirected. Therefore, there is a clear need to build RA solutions over a new, practical link metric that accurately characterizes links in MIMO environments.

In a closed-loop RA model, the receiver's insight into channel conditions is used to compute the transmission rate. A feedback mechanism should therefore be incorporated into the design. In fact, the 802.11n standard supports an explicit feedback system in *MCS Request* and *MCS Feedback* [1]. By exploiting this standard-compliant feedback mechanism, accurate receiver-based RA solutions can be designed for 802.11n MIMO environments.

The state of the art for RA in 802.11n calls for a standard-compliant, closed-loop solution that accurately exploits the new features in 802.11 MIMO environments. Therefore, the RA solution must adopt a link metric that accurately characterizes

MIMO link performance. We propose such an RA solution, called ARAMIS (Agile Rate Adaptation for MIMO Systems) [22].

ARAMIS is a closed-loop, per-packet RA solution that simultaneously adapts both rate and channel width. ARAMIS incorporates a measurement-based, 802.11n link predictor in its design. Given the current channel conditions, the link predictor estimates Packet Reception Rate (PRR) for the given link at all supported rate and bandwidth combinations. Using this information, ARAMIS then selects the best operating point, and sends the feedback to the transmitter using a standard-compliant mechanism. To characterize MIMO link performance and capture channel conditions, we develop a practical link metric called *diffSNR*, which we then use in ARAMIS. *diffSNR* provides a good balance between implementability and accuracy as input to the link predictor. Through testing in a variety of environments, we show that our link predictor estimates link quality with an accuracy of at least 95.5%.

We implemented ARAMIS and evaluated it on a 15 node testbed [22]. We compared ARAMIS to leading RA solutions for 802.11n, namely Ath9k [112], Minstrel HT [25], and RAMAS [76]. We evaluated the solutions under various scenarios, including interference, mobility, and hidden nodes. We demonstrated that ARAMIS is robust, consistently performs well and outperforms existing solutions, with an average of 0.5 fold and up to a 2.87 fold increase in throughput compared to its best

competitor, RAMAS, and an average of 3.85 and up to a 10 fold increase compared to Ath9k.

In addition to comparing ARAMIS to leading RA solutions for 802.11n, we evaluate the absolute performance of ARAMIS compared to an ideal solution, via a trace-driven simulation, and provide a detailed analysis of the impact of ARAMIS's design features on network performance. The packet traces for the simulation were collected in an 802.11n testbed under varied network conditions. This comparison against an ideal solution allows us to gauge how closely ARAMIS approximates a performance upper bound. We find that ARAMIS closely approximates an ideal solution by taking advantage of per-packet processing. Per-packet RA enables quick and fine adjustments to varying channel conditions and allows the exploitation of narrow windows of higher bandwidth opportunities. Ideally, we would compare ARAMIS to an existing solution in literature that uses CSI to perform rate adaptation [40]. However, we are unable to implement CSI-based RA solutions, since our chipsets do not support CSI. We believe, however, that an ideal solution would provide a more accurate upper bound, since it is implementation-independent.

We evaluate the deployability of our solution, and our evaluations reveal that the link predictor component of ARAMIS exhibits relatively consistent behavior for different packet sizes and rates. Not surprisingly, the performance of aggressive modulations is more difficult to predict, and this is translated into higher PRR prediction

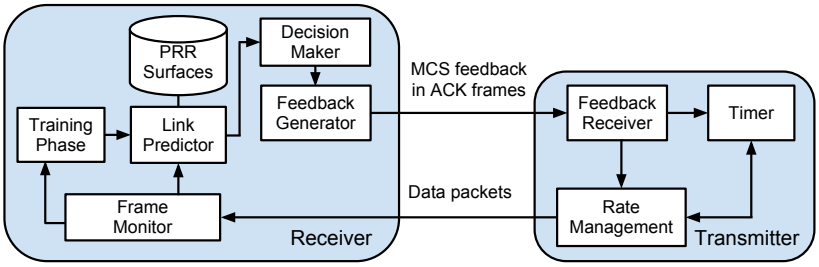


Figure 6.1: Block diagram of ARAMIS.

errors. Furthermore, PRR prediction errors increase when spatial multiplexing is used. Spatial multiplexing relies on the presence of independent propagation paths to successfully decode the transmitted spatial streams. However, accurately identifying independent paths requires costly CSI. ARAMIS's link predictor forgoes the cost of CSI while maintaining a reasonable level of accuracy, with an average absolute error in PRR predictions of 12%, and by adding a training mechanism, errors fall to 5.8%.

Finally, we provide a detailed evaluation of the performance of SNR in 802.11n MIMO environments. Though the inaccuracies of SNR have been identified in prior work [40,93], we show that SNR measurements are still useful to assess the channel, and in fact, SNR forms a key component of our approach. Through our detailed evaluation of SNR behavior, we come to understand the impact of different 802.11n features, namely spatial multiplexing, spatial diversity, and channel bonding, on its

performance. This insight also allows us to identify *diffSNR* as an alternative and practical link metric for 802.11n links.

This chapter is organized as follows. We first present an overview of the design of ARAMIS in Section 6.2. Section 6.3 evaluates the efficacy of SNR and our metric *diffSNR* in 802.11n MIMO environments, where we also detail the implementation cost of CSI. We present our link predictor in Section 6.4, and evaluate its prediction accuracy. Section 6.5 discusses ARAMIS's rate selection algorithm and its components. We then evaluate ARAMIS and compare it to existing solutions under both a simulation and testbed environment in Section 6.6. Related areas of research are discussed in Section 6.7. Finally, we conclude in Section 6.8.

6.2 Overview of ARAMIS

ARAMIS is a closed-loop RA solution for 802.11n MIMO environments. In the design of ARAMIS, we identify three important, high-level components. The first component is a link metric that can be used to accurately characterize MIMO link performance. An existing MIMO link metric, called *effectiveSNR*, is one option [40]. However, *effectiveSNR* is built using CSI, and CSI is supported by few devices due to its costliness [18]. We therefore choose to develop a new, practical, MIMO link

metric for systems where CSI is not available. We call this new metric *diffSNR*, and it provides a good balance between implementability and accuracy.

The second component is a mechanism that can accurately predict the Packet Reception Rate (PRR) of a link for any MCS and bandwidth combination, which we refer to as the *link predictor*. To predict PRR, the *link predictor* uses PRR performance models from the adopted link metric. The *link predictor* and link metric together form the backbone of the third and main design component, the *rate selector*. Based on current channel conditions which are determined using the link metric, and the corresponding PRR values computed using the *link predictor*, the *rate selector* finds the best operating rate and bandwidth with high accuracy. Since ARAMIS is a closed-loop RA solution, the *rate selector* also needs to implement a standard-compliant feedback mechanism.

Fig. 6.1 depicts the specific components in the design of ARAMIS and the corresponding communication flow between a transmitter and receiver pair. The primary functionality of ARAMIS is implemented at the receiver.

The first interface into ARAMIS's *rate selector* is the *Frame Monitor*. The *Frame Monitor* maintains updated information on channel conditions by measuring the link metric from existing data traffic. Current channel status information is used as input to the *Link Predictor*, which estimates the PRR of the link for all supported MCS and bandwidth combinations. The *Link Predictor* is measurement-based, and

therefore must first access data storage to obtain the measured PRR values. To improve the accuracy of predictions, the *Link Predictor* goes through a *Training Phase* that corrects errors in predicted PRR values in real-time. The *Decision Maker* then takes the PRR predictions from the *Link Predictor*, and based on some performance model, selects the best operating point. Using a standard-compliant mechanism, the *Feedback Generator* encapsulates information on the best possible rate in ACK frames to be sent to the transmitter. However, in the case when the feedback is not received, for example due to lack of active traffic, a backup *Timer* is implemented at the transmitter to reset the operating rate.

In the following sections, we describe the main features of ARAMIS, namely the adopted link metric, the *link predictor*, and the *rate selector*.

6.3 802.11n MIMO Link Metrics

We are first motivated by the need for a new metric by identifying the limitations of a commonly used and accessible link metric, RSSI (Received Signal Strength Indicator), in predicting link quality in 802.11n MIMO environments. We measure link quality or performance in terms of Packet Error Rate (PER, where: PER $= 1 -$ PRR). We then present *diffSNR* and examine how it can be used together with RSSI to accurately reflect performance.

For legibility, we present a subset of our results that best represents the patterns in the behavior of RSSI and *diffSNR*. We conduct all experiments for both 20MHz and 40MHz channels, and we discuss our observations for three MCS indices that cover robust (MCS 8), intermediate (MCS 12), and aggressive (MCS 15) PHY rates. For each MCS, we send 5,000 1kB UDP datagrams from the AP to its client.[1]

6.3.1 The Limitation of RSSI

RSSI, defined as the signal to noise ratio (SNR) in dB, has traditionally been used to represent the quality of a link [13]. The existing models that map RSSI to performance show that a link's PER is 1.0 for sufficiently low RSSI and then steeply drops to 0.0 as RSSI increases beyond a threshold value [3].

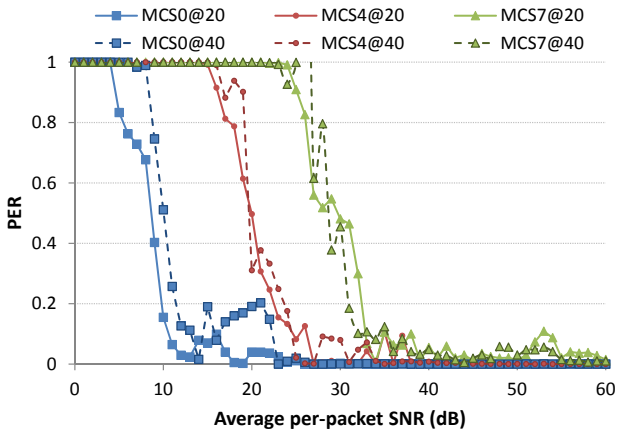
Fig. 6.2 plots PER vs SNR averaged over 50 links in our testbed. Fig. 6.2(a) plots the values for one transmit stream and Fig. 6.2(b) for two streams. As RSSI increases, we expect PER to drop since the receiver can better decode the received signal. Fig. 6.2, however, shows that this is not necessarily the case. When we compare Figs. 6.2(a) and 6.2(b), we observe irregular behavior particularly for aggressive modulation schemes with spatial multiplexing (MCS 12 and 15). PER does not converge to 0 for high SNR and surprisingly in Fig. 6.2(b), performance degrades for SNR $>$ 55dB.

[1] See Section 6.6 for testbed details.

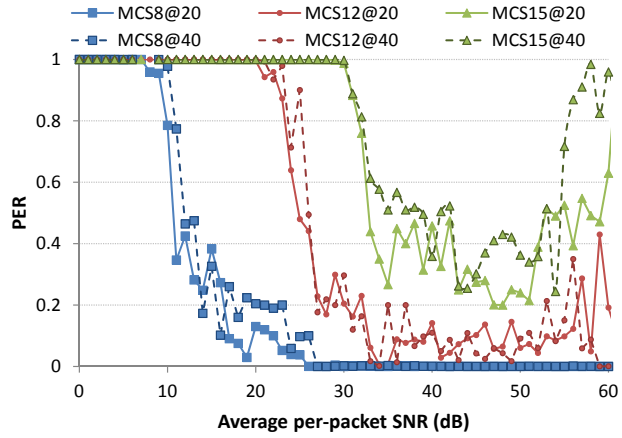
There are two explanations to this behavior. High SNR values are achieved when the output power is high and/or when the propagation losses are low due to the close proximity of the transmitter/receiver pair in the absence of obstacles. The combination of OFDM and high order amplitude modulations (such as 64-QAM used in MCS 13 to 15) is prone to high peak-to-average ratios: high peaks cause the power amplifiers to move toward saturation [5], exhibiting non-linear behavior that produces inter-modulation distortion. The other explanation is that the presence of a dominant path between a transmitter/receiver pair, such as when the nodes are in direct line-of-sight, increases the Rician K-factor and the channel becomes increasingly correlated in space. This hampers the utilization of spatial multiplexing [120].

Fig. 6.2 also shows that SNR is a poor indicator of link quality for different channel widths. For the same SNR, a 40MHz channel suffers a higher PER. Wider transmissions are more likely to suffer from frequency selective fading, which causes SNR variations across the OFDM subcarriers, and PER is dominated by the lower SNR carriers. A 40MHz channel, therefore, not only requires a stronger transmission power to achieve the same SNR [7] but also a higher SNR to provide the same PER.

Figs. 6.3 and 6.4 plot per-link PER vs SNR for all testbed links and bandwidths. On the Y-axis, each point represents PER measured over 5,000 frame transmissions in one particular link. On the X-axis, each point is the average SNR of the received

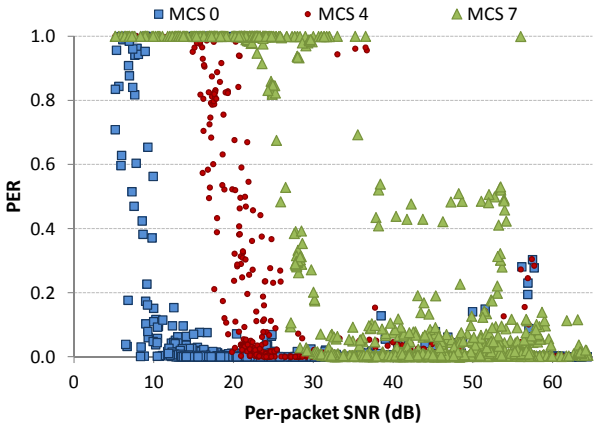


(a) One transmit stream.

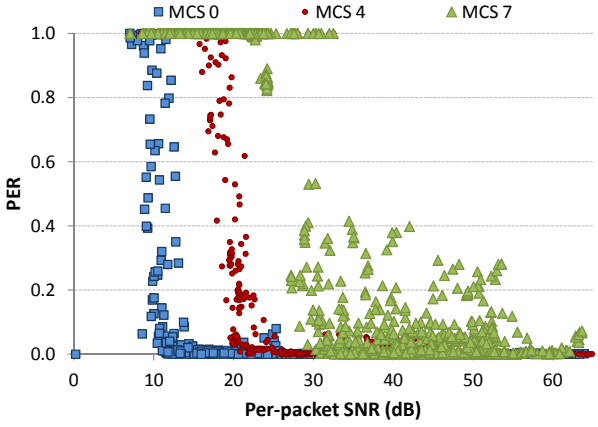


(b) Two transmit streams.

Figure 6.2: Average PER and per-packet SNR over testbed links.

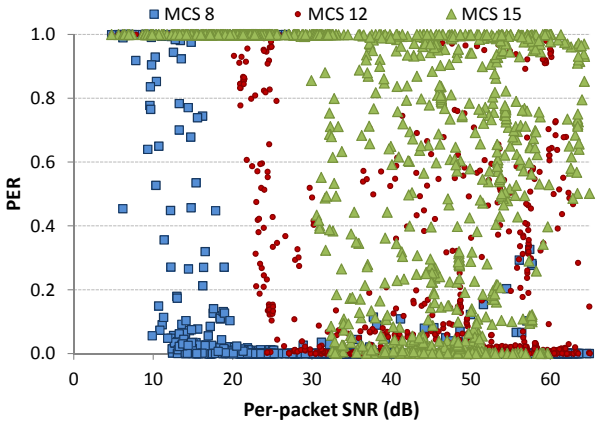


(a) 20MHz wide channel.

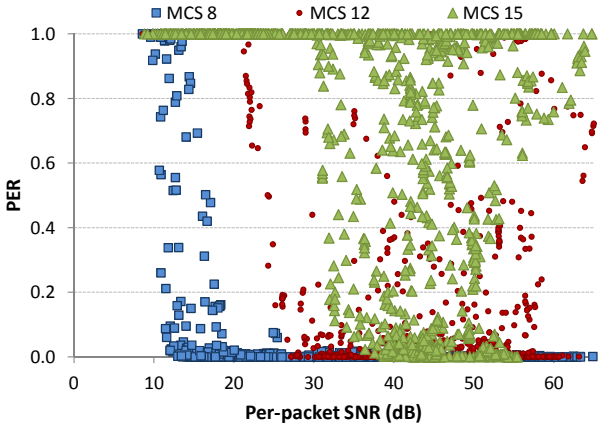


(b) 40MHz wide channel.

Figure 6.3: Per-link PER vs SNR measurements for one stream.



(a) 20MHz wide channel.



(b) 40MHz wide channel.

Figure 6.4: Per-link PER vs SNR measurements for two streams.

stream of packets. Fig. 6.3 shows results for a single stream, while Fig. 6.4 for two streams.

Figs. 6.3 and 6.4 show that RSSI is a reliable metric when robust MCS modes are used that exploit spatial diversity. For example, the transition region for MCS 0 is only 3-4dB wide. For a given link at MCS 0, if the measured SNR is below 6dB, the link is infeasible (PER ≈ 1) and, if the SNR is above 10dB, it is feasible (PER ≈ 0). However, for SNR values between 5 and 10dB, the feasibility of a link is uncertain; some links yield an excellent performance with an SNR of 6dB, while others are not feasible with a higher SNR of 10dB. This uncertainty is amplified as more aggressive modulations are used, where the transition region between a feasible and an infeasible link becomes wider. For example, when both spatial multiplexing and moderate or fast PHY rates are used (e.g. MCS ≥ 12), the transition region could be as wide as 35dB! In such cases, RSSI alone does not provide sufficient information to assess the feasibility of a link. This result is consistent with previous work [40, 93].

6.3.2 The Cost of Channel State Information (CSI)

CSI describes the current channel conditions with fine granularity.[2] It consists of the attenuation and phase shift for each spatial stream to every receive antenna, for every OFDM subcarrier (52 subcarriers for a 20MHz bandwidth and 114 for

[2]Here, we refer to the full CSI matrix and not the coarse grained CSI provided by Intel 802.11n chipsets [40].

a 40MHz bandwidth in 802.11n). Measuring a complete and timely CSI for all possible MIMO channel configurations requires excessive sampling overhead [18].

In some implementations, successful decoding of a data packet is required to compute CSI [40]. Additionally, for a $T \times R$ MIMO system of bandwidth W, a packet is required to be sent using T transmit antennas over a bandwidth W, and received over R receive antennas to obtain the complete $T \times R \times W$ CSI matrix. For example, a 3×3 MIMO system requires seven samples: a single stream MCS requires three probes, one for each transmit antenna; a two stream MCS requires three probes, one for each combination of two transmit antennas; finally, a three stream MCS requires a single probe for all three transmit antennas.

Communicating the computed CSI matrix in a feedback packet also consumes significant bandwidth overhead. The size in bytes of a feedback packet with complete, noncompressed CSI is 1.029KB for a 20MHz channel and 2.095KB for a 40MHz channel [1]. Based on channel coherence time [8], CSI at the transmitter needs to be updated at least once every 50ms. CSI feedback as a result consumes 160.64Kb/s to 335.2Kb/s respectively. In a per-packet RA implementation where CSI needs to be updated frequently, the bandwidth consumed by CSI feedback quickly becomes a significant overhead.

Complete CSI is clearly expensive to obtain and communicate, and therefore its applicability to a per-packet RA solution, particularly in dynamic environments

where timely channel information is necessary, is limited. We therefore need to identify an alternative MIMO link metric in the design of ARAMIS.

6.3.3 Differential SNR (*diffSNR*)

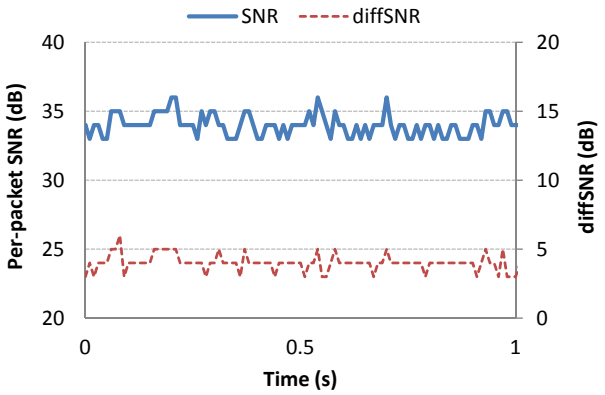
It is clear that RSSI alone does not accurately capture the factors that cause the variability in 802.11 channels. Frequency selectivity due to multipath is one major factor whose effects are only captured using OFDM per-subcarrier SNR information [40]. Antenna correlation, or spatial selectivity, is another factor [98]. Both factors, however, require costly CSI which is supported by only very few devices [18]. For devices that do not support CSI, we develop a practical metric, called *diffSNR*, by using the channel metrics available to us in all commodity MIMO devices. We now show how we can use *diffSNR* to accurately reflect channel quality in 802.11n networks.

Multipath propagation in wireless environments produces constructive and destructive interference at the receiving antennas [104]. The resulting signal combination varies at different locations, a concept referred to as spatial selectivity. MIMO systems take advantage of these multipath phenomena to improve performance.

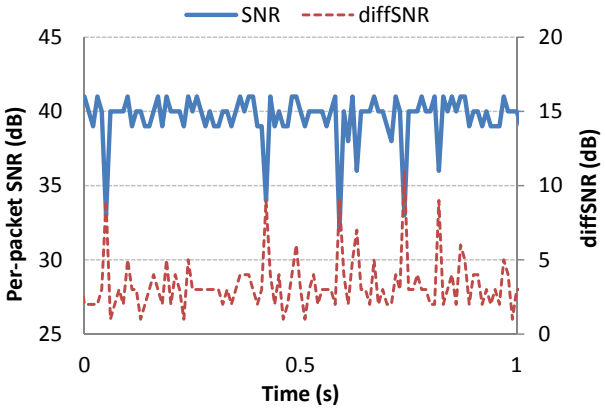
When received signals combine destructively in a process called *selective fading*, SNR can degrade and will reliably indicate a lossy link. Since per-packet SNR is the linear sum of all per-antenna measurements, if only a portion of the antennas

experience fading, the reported SNR may be high even though the link could be lossy. Reported SNR does not reflect the extent of selective fading. We therefore argue that knowledge of the SNR combined with the per-antenna SNR provides us with some added insight, which can be used to predict the link performance with greater accuracy. We henceforth define the difference between the best and the worst SNR at any of the receiver's antennas as *diffSNR*.

After analyzing real-time traces of RSSI and *diffSNR* in different scenarios, we observe that *diffSNR* does not depend significantly on (i) the transmitter's output power (*diffSNR* varies less than 6% when varying tx power of a given link), (ii) the MCS used (*diffSNR* varies less than 2% when varying MCS for a given link), or (iii) the channel width. On the other hand, *diffSNR* shows a clear dependency on the environment: factors such as rich scattering, dynamic/static positioning, line-of-sight, and obstacles. Fig. 6.5 provides two paradigmatic examples of the real-time evolution of SNR and *diffSNR*. We find that a static scenario exhibits fewer variations, as shown in Fig. 6.5(a), while a more dynamic environment is reflected in a wider dispersion of the measured *diffSNR* which exhibits frequent peaks, as depicted in Fig. 6.5(b). A peak in *diffSNR* can occur when RSSI increases and a subset of the antennas receive constructive interference. However, we observe that high diffSNR peaks are often (80% of the time) caused by some of the antennas suffering from fad-



(a) Static scenario (night run) and 20MHz channel.



(b) Dynamic scenario (during office hours) and 20MHz channel.

Figure 6.5: Real-time evolution of per-packet SNR and *diffSNR*.

ing; that is, there is a negative correlation between RSSI and *diffSNR*. This behavior is also deduced in Fig. 6.5(b).

Given the predictable behavior of *diffSNR* and its correlation to RSSI, we next examine the implications of the (SNR, *diffSNR*) relationship and how it can be used to determine link quality or performance in terms of PER.

6.3.4 *diffSNR* and Packet Error Rate (PER)

For links with similar RSSI, we find that *diffSNR* can be used to characterize their performance differences. We illustrate this behavior in Fig. 6.6 using three representative links. We evaluate their PER vs SNR relationships using spatial multiplexing (MCS 12 and 15) and a 40MHz channel.

Link 1 successfully transmits packets (PER $<$ 0.02) using MCS 12; for MCS 15, there is a clear transition around 34dB SNR. Although Link 2 has similar RSSI values to Link 1, it clearly exhibits worse performance: for MCS 12, PER increases rapidly for SNR $<$ 30dB and MCS 15 remains lossy until SNR $>$ 44dB. The difference between Link 1 and 2 can be explained with *diffSNR*: for Link 1, we measure an average *diffSNR* of 1.82dB, with a standard deviation of 0.30, while for Link 2, the average *diffSNR* is 9.46dB, with a standard deviation of 0.37. Link 3 displays the worst performance, showing an average *diffSNR* of 13.41dB. This link does not exhibit a clear transition for MCS 12 and never works for MCS 15. We can explain this behavior with the dispersion of its measured *diffSNR* values with a standard deviation of 0.97.

Our analysis reveals the dependency of performance on RSSI and *diffSNR* together. Since robust modulations are less affected by fading, variations in *diffSNR* will be more clearly reflected on the performance of aggressive modulations. Sim-

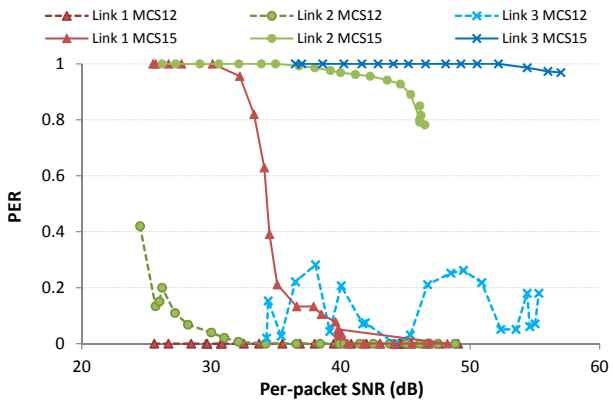
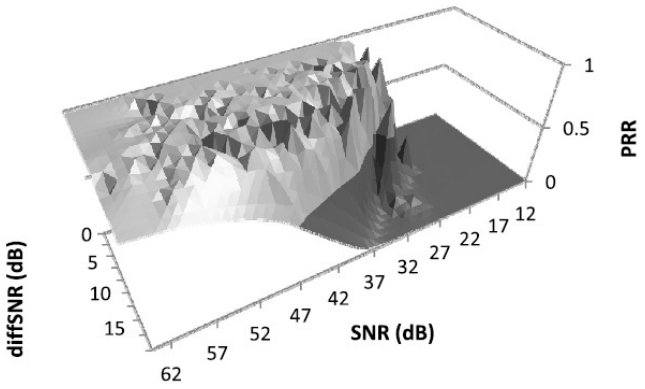


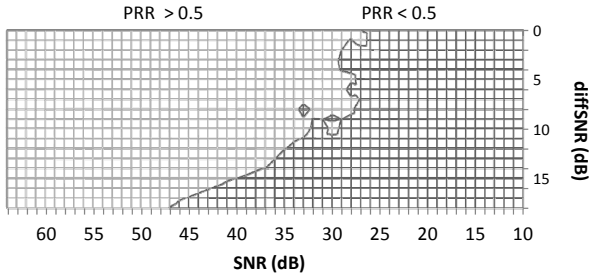
Figure 6.6: PER vs per-packet SNR for three links (40MHz channel).

ilarly, *diffSNR* variations will have little impact on links with high SNR, but this impact will increase as SNR decreases. We believe this dependency is intriguing and leave further analysis of potential correlations to spatial or frequency selectivity to future work. Fig. 6.7 is a representative graph that effectively exemplifies this dependency.

Fig. 6.7 plots the measured PRR as a function of average per-packet SNR and *diffSNR* for a given MCS and bandwidth combination. We note that the dataset in Fig. 6.7 combines values obtained from real measurements with interpolated points. Fig. 6.7(a) shows that the PRR(SNR, *diffSNR*) relationship yields well-behaved surfaces that allow us to predict the PRR of a link for a given MCS and bandwidth. For example, as shown in the projected image in Fig. 6.7(b) with an SNR of 32dB, a link with *diffSNR* below 5dB performs well. However with a *diffSNR* above 10dB, the



(a) PRR(SNR,*diffSNR*) surface.



(b) 2D projection with isoline at PRR = 0.5.

Figure 6.7: PRR as a function of packet-SNR and *diffSNR* for MCS 7 and 20MHz channel.

PRR falls to almost 0. Next, we describe how we utilize these surfaces in the design of our link predictor.

6.4 A Measurement-Based Link Predictor

A link predictor accurately estimates the PRR of a link for all MCS and bandwidth combinations. We now describe the methodology we use to build such a

predictor, and demonstrate how it accurately predicts PRR. In case of errors, we introduce a low-overhead training mechanism to improve accuracy.

6.4.1 Methodology

We design our predictor as the synthesis of the measurement-based, pre-computed PRR(SNR, *diffSNR*) surfaces for all MCS and bandwidth combinations. SNR and *diffSNR* measurements, along with the operating MCS and bandwidth of a link, are the input parameters the predictor uses to identify the corresponding expected PRR for that link. It is important to note that our testbed provides us with SNR data for the control (i.e. primary) 20MHz channel and when channel bonding, the extended 40MHz channel. Predictions for 20MHz links can be made from measurements under 40MHz links, but not vice versa [20,21]. Therefore, our predictor builds separate PRR surfaces for both 40MHz and 20MHz channels for each MCS.

To gather sufficient data to pre-compute and build the PRR(SNR, *diffSNR*) surfaces for each MCS and bandwidth combination, we measure PRR(SNR, *diffSNR*) over all 50 testbed links while varying the transmit power from 0dBm to the maximum allowed power. As for the (SNR, *diffSNR*) data points that do not have measured values, we fill them by interpolation, using the nearest measured data points.

Intuitively, PRR depends on the packet length. Hence, packet length should be accounted for to predict PRR. The PRR for any given packet size can be roughly

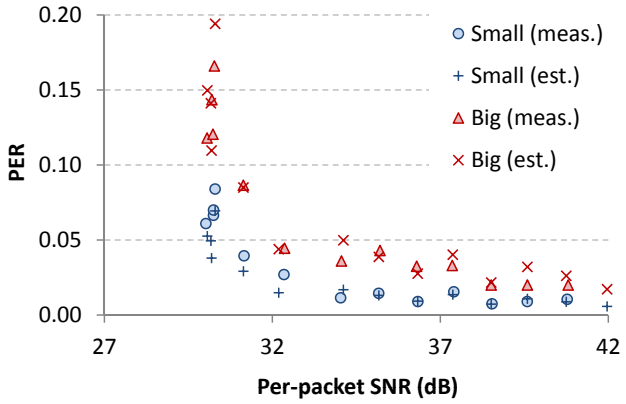


Figure 6.8: Measured and estimated PER vs SNR for 300B (Small) and 1500B (Big) packets.

estimated from the PRR we measure for 1,000 byte packet size using the equation: $P_x = \left(\sqrt[L_{1000}]{P_{1000}} \right)^{L_x}$, where P_x is the PRR for a packet of length x bytes, and L_x is the length in bits of a frame carrying an x byte packet [**?**]. For different packet sizes, our measurements show that the transition regions for links from high quality to lossy do not exhibit a noticeable difference. Fig. 6.8 plots both the estimated PER, following the P_x equation above, as well as the measured PER for 1500B (Big) and 300B (Small) packet transmissions for a given link using MCS 12. We observe that, as expected, larger packets show higher PER; however, the impact of packet size on the feasibility of a link is still negligible. This result indicates that transition regions do not depend on frame size, and thus we do not add frame size as an additional dimension.

6.4.2 Prediction Accuracy

The link predictor is a pre-computed matrix, with dimensions defined by the number of supported MCS, bandwidths, and the range of expected SNR and *diffSNR* values. To evaluate our predictor, we build two $6 \times 70 \times 20$ matrices for MCS 0, 4, 7, 8, 12 and 15, with SNR values from 0 to 69dB and *diffSNR* values from 0 to 19dB, with 1dB precision.[3] Our complete predictor consists of two $16 \times 70 \times 20$ matrices, and is used in future sections. The reduced matrix consists of 40% of measured values (the remaining 60% are interpolated). We show that interpolation has no significant impact on the prediction accuracy.

We evaluate the accuracy of our measurement-based, pre-computed link predictor by comparing the predicted PRR values against the measured values for two different groups of transmitter/receiver pairs. The first group consists of nodes located in the same environment where the data for the predictor was collected. The second group consists of a set of laptops placed in two different off-campus small office/home office environments as well as in an outdoor environment, on a rooftop free of obstacles with direct LoS between nodes placed 20m apart. We include this second group to evaluate the utility and accuracy of the proposed predictor in unfamiliar and dissimilar environments.

[3]The distribution of *diffSNR* in all tested environments lie below 19dB.

For the first group of nodes located in a familiar environment, the average absolute error in PRR predictions, computed as the difference between the measured PRR and the predicted PRR, is only 4.8%. This error rate increases for high order modulations (up to 11% for MCS 15) since these modulations show a higher degree of uncertainty. Although the absolute error may be relatively high for some MCS indices, we reliably predict link feasibility with a 96.1% accuracy.[4] As for the second group of nodes in new environments, the average absolute error in PRR predictions is 12% and the accuracy in feasibility predictions is 88.1%. These results show the importance of a calibration or training mechanism. To increase the prediction accuracy, we include the error of previous measurements in the new PRR predictions such that:

$$PRR_k^{m,B} = PRR(m, B, SNR_k, diffSNR_k) + E_{k-1}^{m,B} \tag{6.1}$$

where $PRR_k^{m,B}$ is the predicted PRR for MCS m and bandwidth B; SNR_k, and $diffSNR_k$ are the currently measured RSSI and *diffSNR* values; and $PRR(w, x, y, z)$ returns a PRR value from the predictor using the input parameters. Finally, $E_{k-1}^{m,B}$ is the error in previous predictions for the same MCS and bandwidth, where $0 \leq E_{k-1}^{m,B} \leq 1$. We track the error by computing $E_k^{m,B}$ as an exponential moving average with α = 0.9. Our α is large to give more weight to recent error samples, since the long-term mean error in PRR predictions is close to 0.

[4] We consider a link feasible for a given MCS if PER < 0.5.

We re-evaluate our results in the new environments using Eq. 6.1. Fig. 6.9(a) computes the average absolute error in PRR predictions, for all tested MCS indices. On average, the error in our improved PRR predictions lies below 5.8%. Fig. 6.9(b) shows that link feasibility predictions improve to a 95.5% hit rate.

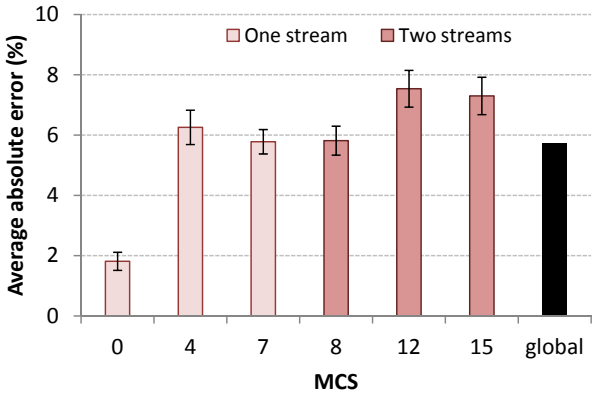
A detailed evaluation of PRR predictions in these two new environments reveals the same patterns observed in the lab measurements. First, performance of aggressive modulations is more difficult to predict, and this is translated into lower feasibility prediction rates and higher PRR prediction errors. Second, PRR prediction errors increase when spatial multiplexing is used (4.6% average absolute error for one stream vs 6.9% for two). Finally, the PRR of a 20MHz channel can be predicted with slightly greater accuracy than a 40MHz channel (96.0% feasibility prediction hits for 20MHz channels vs 95.1% hits for 40MHz).

Spatial multiplexing, aggressive modulations, and wider 40MHz channels are all 802.11n features that achieve higher data rates at the risk of greater susceptibility to loss and changes in environment conditions. For example, spatial multiplexing relies on the presence of independent propagation paths to successfully decode the transmitted spatial streams. Further for the same link, a 40MHz channel has a lower SNR than a 20MHz channel and is more susceptible to frequency selectivity [7]. Accurately reflecting such detailed environment conditions however, such as the number of independent paths and frequency selectivity, is achieved using fine-grained CSI

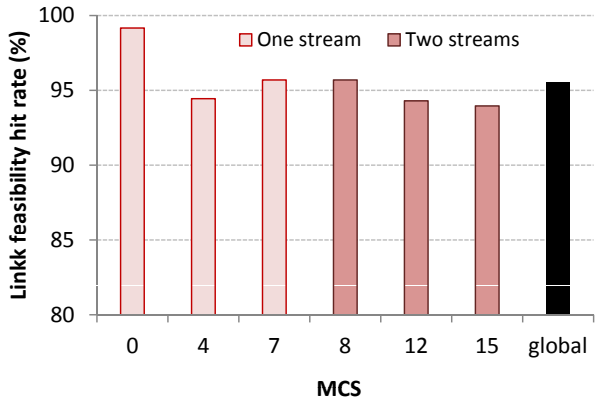
alone. ARAMIS's link predictor forgoes the high cost of fine-grained CSI for ease of implementation, by using coarse-grained information on channel conditions, while maintaining a reasonable level of accuracy in predicting environment conditions.

Recall that we interpolate to fill the gaps in a PRR(SNR, *diffSNR*) surface. We observe that regardless of whether the predictions come from interpolated or measured values, the predictor accuracy remains the same. For the measurements conducted in unfamiliar environments, we predict 71% of the indoor links from measured values while the remaining 29% from interpolated values. For outdoor links, the proportion is 61 measured to 39% interpolated. For the familiar environment, interpolated values are not used. We note that the ratio of predictions that are made from interpolated versus measured values reflects how accurately we are capture similar (SNR, *diffSNR*) channel conditions between environments. It is therefore not surprising that the outdoor environment has a greater number of predictions from interpolated values, as it exhibits different multipath characteristics from an indoor environment, where the PRR(SNR, *diffSNR*) surfaces were built.

We have built a mechanism capable of accurately predicting PRR for all MCS and bandwidth combinations for a given link (SNR, *diffSNR*). This accuracy enables us to include our predictor in a rate selection mechanism.



(a) Performance of link PRR predictions (99% confidence intervals).



(b) Performance of link feasibility predictions.

Figure 6.9: Evaluation of the link predictor in new environments.

6.5 Rate Selector

The *rate selector* is the final and main design component of ARAMIS. An effective rate selector in a closed-loop, 802.11 RA model identifies changes in environment conditions and responds with the appropriate rate using a standard-compliant feedback method. To achieve these goals, we now describe how we combine our knowledge of our link metric, in this case (SNR, *diffSNR*), and the *Link Predictor* in the design of an effective rate selector. We use the terminology illustrated in Fig. 6.1.

Algorithm 3 ARAMIS(SNR, *diffSNR*)

Output: 1) MCS m; 2) Channel width B;

1: **if** $newPacket =$ true **then**
2: (SNR$_{avg}$, *diffSNR*$_{avg}$) $\leftarrow$ update-moving-average(SNR, *diffSNR*)
3: **if** exception(SNR, *diffSNR*) $=$ true **then**
4: (m,B) $\leftarrow$ decision-maker() $\leftarrow$ link-predictor(SNR$_{avg}$,*diffSNR*$_{avg}$)
5: **end if**
6: **end if**

6.5.1 Frame Monitor

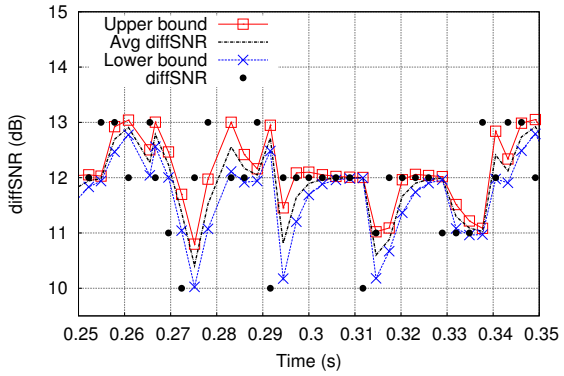
The first step of a rate selector is to identify changes in channel conditions. This step is necessary to determine when an alternative rate might be appropriate. We have verified the accuracy of (SNR, *diffSNR*) in predicting link quality. We now

describe how we monitor the behavior of per-packet (SNR, *diffSNR*) in real-time, using existing active traffic, to identify changes in channel conditions.

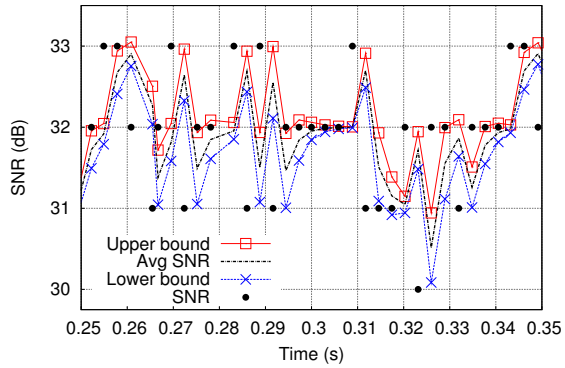
Fig. 6.5 depicts the evolution in per-packet (SNR, *diffSNR*) over time for a given link. Over a short period of time, (SNR, *diffSNR*) can fluctuate rapidly. To identify when changes in (SNR, *diffSNR*) could reflect a change in channel conditions, we apply an exponentially weighted moving average approach. ARAMIS stores (SNR, *diffSNR*) for every packet received and computes their moving average (SNR_{avg}, $diffSNR_{avg}$). We maintain moving averages not only for the average (SNR, *diffSNR*) values, but also for their standard deviation (SNR_{sd}, $diffSNR_{sd}$). ARAMIS initiates lookups to the link predictor if the current (SNR, *diffSNR*) lies outside of the range specified by $SNR_{avg} \pm SNR_{sd}$. The same conditions apply for *diffSNR*.

6.5.2 Decision Maker

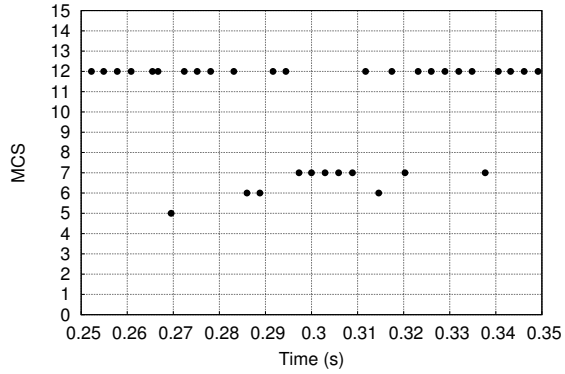
Our *rate selector* uses a link's current channel conditions, reflected through the link metric, as input arguments to the *Link Predictor*, in this case using (SNR_{avg}, $diffSNR_{avg}$). The *Link Predictor* determines accurate PRR estimates for all supported MCS and bandwidths for that link, as described in Section 6.4. The role of the *Decision Maker* is to use this information to select the MCS and bandwidth configuration that yields the highest throughput. One model would be to select the configuration with the highest expected throughput. The computation of the expected throughput,



(a) Variation of *diffSNR* with time.



(b) Variation of SNR with time.



(c) Selected rate based on *diffSNR*, SNR values from previous packet measurements.

Figure 6.10: Depiction of ARAMIS measurements and behavior. We show that ARAMIS responds to changes in (SNR, *diffSNR*) conditions by modifying the MCS and bandwidth when necessary. In this case, the link always chooses a 40MHz channel.

Table 6.1: HTC subfields that support receiver-based RA.

MRQ	MCS feedback request
MSI	MRQ sequence identifier
MFB	MCS feedback
CBF*	AP Channel bonding friendly
MIR*	Client MCS index request
CW*	Client channel width request

*: Bits allocated to support channel width feedback

however, requires a foreknowledge of the packet size implemented at the transmitter [20], which is not available at the receiver. Furthermore, this approach adds significant overhead to the computation of the appropriate rate.

We adopt a simple yet effective approach. Our model selects the MCS and bandwidth combination with the highest PHY bitrate from a reduced set of combinations whose predicted PRR is above a threshold. By adjusting this threshold, ARAMIS has the flexibility to adapt to environments with varying PER tolerances.

Fig. 6.10 demonstrates the behavior of ARAMIS in real-time, as described in Algorithm 3. In Fig. 6.10(a) and (b), we plot the instantaneous values, moving averages, and upper and lower bounds for our link metrics, both SNR and *diffSNR*. Fig. 6.10(c) depicts how ARAMIS changes MCS on a per-packet basis based on the correlated (SNR, *diffSNR*) values, where ARAMIS selects the MCS with the highest bitrate from those MCS that achieve a PRR above a given threshold for the current channel conditions. The corresponding bandwidth graph is not shown as ARAMIS always opts for a 40MHz channel for this link. Although not depicted, ARAMIS opted for a 20 MHz channel, particularly for the weakest links in our evaluation.

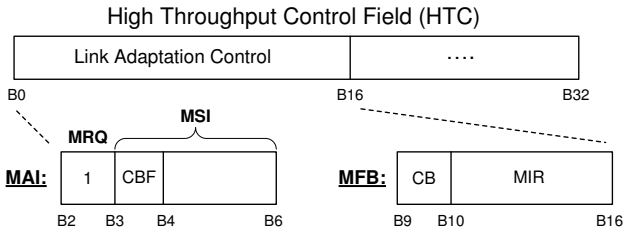


Figure 6.11: 802.11n compliant MCS feedback system.

6.5.3 Training Phase

To improve the accuracy of predicted PRR values for all MCS and bandwidth combinations, a training mechanism is performed on-the-fly using the statistics of received frames, which does not incur any extra overhead. ARAMIS measures the link's actual PRR by dividing the number of received frames with the Retry flag set to 0, by the total number of frames sent, the latter computed using frame sequence numbers. If aggregation is enabled, more precise PRR estimation could be provided by inspecting the bitmap field present in the Block ACK. ARAMIS then uses this measured PRR to update $E_k^{m,B}$ values in Equation 6.1.

6.5.4 Feedback Generator

We have discussed how ARAMIS identifies an appropriate rate given the current channel conditions. This rate, however, should be sent as feedback to the transmitter using a standard-compliant mechanism. To fully exploit variations in a MIMO

channel, the 802.11n standard supports MCS feedback (MFB) in link adaptation [1]. MFB is a subfield of the *HT Control field* (HTC). HTC is a 4B optional field added to control packets (such as ACKs and Block ACKs).

Fig. 6.11 shows the HTC field with its corresponding link adaptation control field, where the subfields are described in Table 6.1. We propose utilizing the unused fields and creating subfields that control bandwidth feedback. These added subfields allow ARAMIS to operate in conjunction with a channel management solution [21], where the CBF field set by the AP defines the supported bandwidth in the given WLAN. This allows ARAMIS to make informed channel width decisions using the insight from network layer conditions. For example, if CBF is set to 1 by a channel management approach, the client can request to operate on both a 20MHz and 40MHz channel, which it specifies in the CW subfield, and if CBF is set to 0, the client only operates on a 20MHz channel. It is worth noting that the emerging 802.11ac standard supports such a client-based bandwidth adaptation mechanism, given the maximum supported bandwidth at the AP.

6.5.5 Timer

A transmitter stops receiving feedback when the ARAMIS receiver does not receive transmitter frames. This can happen for two reasons. First, channel conditions at any given time could change drastically such that the PRR for the PHY config-

uration in use suddenly drops to 0. Second, the transmitter may not have traffic to send. In both cases, the communication could be set at the wrong configuration with outdated information, since the transmitter is not receiving feedback to identify the appropriate MCS and bandwidth. This can lead to performance degradation. To mitigate this problem, we use a timer at the transmitter, whereby if feedback packets are not received before the timer expires, the MCS is set back to a reliable rate, MCS 8, then MCS 0 after a consecutive timeout, at the same bandwidth.[5] Our results show that ARAMIS's per-packet rate adaptation is able to rapidly recover from this MCS reset.

6.6 Performance Evaluation

We evaluate ARAMIS, first using simulation based on packet traces from our experimental platform. We then implement ARAMIS on a real testbed and compare its performance to that of existing RA solutions under various network conditions. The goal of the trace-driven simulation is to evaluate the design choices for ARAMIS, since it gives us the flexibility to reproduce environment conditions while evaluating the performance of various design parameters. Furthermore, the post-processing of

[5]This timer presents a tradeoff: a small timer may hastily fall back to low rates in the presence of severe collisions, and a large timer may prevent ARAMIS from rapidly adapting to degradations in signal quality.

these traces allows us to simulate an optimal (and impractical) algorithm to use as a benchmark for our approach.

In our testbed implementation, we evaluate ARAMIS under various scenarios, including interference, mobility, and hidden nodes. Our goal is to demonstrate the efficacy of ARAMIS in accurately responding to channel conditions compared to other popular 802.11n RA solutions. We measure performance in terms of achieved throughput.

Testbed details: Both evaluation environments are built over our testbed platform that consists of 15 laptops deployed in both an open office and semi-open office environment. Each laptop is equipped with an 802.11n 2×3 MIMO PC card with an Atheros AR5416/AR5133 2.4/5GHz chipset. The AR5416 baseband and MAC processor allow MCS indices 0 to 15. Each laptop runs the Atheros Ath9k device driver that supports 802.11n [112]. We run our experiments on the 5GHz range and verify the lack of background traffic with a spectrum analyzer.

6.6.1 Trace-Driven Simulations

Simulation Environment

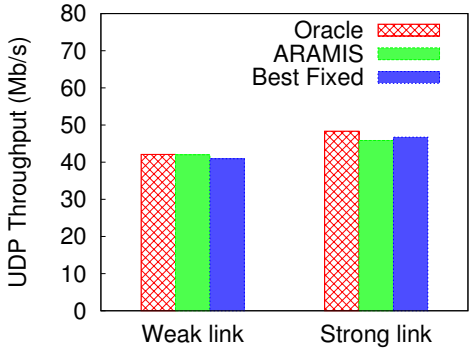
The simulation utilizes packet traces that we collect over our testbed for various links. For each MCS, we send 5,000 1kB UDP datagrams from the AP to its

client at a constant inter-packet delay of 2.7ms. We introduce this delay to avoid issues related to buffer overflow and conditions that restrict our ability to reproduce environment conditions. For the same reasons, we disable packet aggregation. The packet traces are stored at the client and consist of per-packet (SNR, *diffSNR*) values and inter-packet delays, as well as the computed PER, average SNR, and throughput for the entire transmission. We fix the transmit power to $11dBm$, which is the maximum common power level among all MCS. We conduct the above for both 20MHz and 40MHz channels.

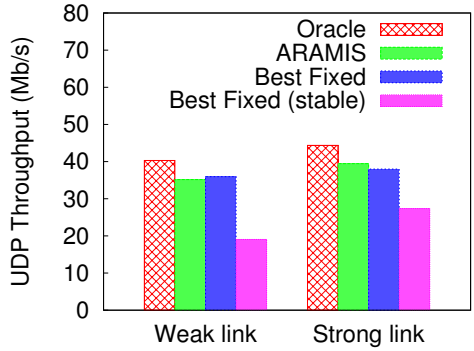
We collect packet traces in interference-free environments as well as controlled interference conditions shown to affect 802.11n performance [20]. For the interference conditions, we introduce an interfering link that operates on either the same or an adjacent channel.

We use the above packet traces as input to our simulator. The simulator is built in custom C and Python. The simulator works by replaying per-packet transmissions using each packet's transmission characteristics, namely its MCS, channel width, delay to the next consecutive packet, and packet loss.

We implement ARAMIS and other solutions, namely *Best Fixed* and *Oracle* in our simulator. *Best Fixed* fixes the MCS that maximizes throughput for the entire simulation run, and serves as a performance baseline. *Best Fixed* differs from ARAMIS in that it does not perform per-packet RA, but rather per-transmission RA



(a) Interference-free.



(b) Interference conditions.

Figure 6.12: Comparison of ARAMIS against other representative solutions.

by selecting the best MCS that maximizes throughput for that transmission. We add *Best Fixed (stable)* to the set of alternative solutions, and it represents the MCS *Best Fixed* chooses under stable, interference-free conditions. Finally, *Oracle* uses the foreknowledge of the performance of each link for every MCS and bandwidth combination to make optimal per-packet RA decisions, and serves as an upper bound for performance. Each simulation is run for 200s of simulation time.

Simulation Results

Figure 6.12 presents the simulation results. Figure 6.12(a) depicts the UDP throughput under interference-free channel conditions for two types of links, where a weak link is unlikely to support high MCS due to weak SNR/*diffSNR*. Figure 6.12(b) shows the result with channel interference. We include *Best Fixed (stable)* to Figure 6.12(b) to show how the best MCS that is selected in interference-free conditions would perform in interference environments. The insight from these results is the importance of *per-packet* rate adaptation in the presence of interference as well as in stable environments, where changes in the channel occur on narrow timescales. *Best Fixed (stable)* performs poorly when interference is introduced. ARAMIS takes advantage of per-packet processing, thus allowing quick and fine adjustments to varying channel conditions. In the presence of interference, however, there are fewer opportunities to take advantage of per-packet RA. As a result, ARAMIS is shown to provide near-optimal performance, similar to that obtained by the best fixed MCS, which is another ideal solution that requires foreknowledge of interference conditions to select the appropriate MCS. With aggregation disabled, all three solutions show little performance differences since they are close to the maximum theoretical throughput. In the next section we show how this PHY adaptation is combined with aggregation, a link layer feature, to leverage the potential of IEEE 802.11n.

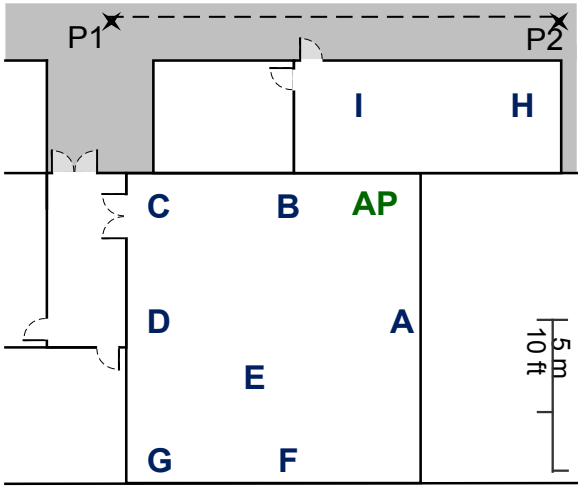


Figure 6.13: Floorplan of our testbed environment.

6.6.2 Testbed Implementation

Testbed Environment

We compare the performance of ARAMIS to that of two widely used open source 802.11n RA solutions, Ath9k [112] and Minstrel HT [25], and RAMAS [76], which was recently shown to be one of the best performing 802.11n RA solutions. We run RAMAS using the implementation made available by its authors. RAMAS is a credit-based system that divides the features of 802.11n RA into two groups: a modulation group and a group that includes the number of streams and bandwidth. Each group follows a different credit system and is adapted independently of the other. Minstrel HT and Ath9k both use random sampling to find the best MCS. Minstrel HT, however, includes MCS with different bandwidths in its sampling group. Ath9k

does not have a mechanism for enabling channel bonding, and to ensure a fair comparison, we set Ath9k's bandwidth to 40MHz to allow it to exploit higher data rates. Ath9k switches to a 20MHz channel when the PER is high. Other schemes select channel width based on their algorithm, and independently of the rate.

We evaluate the RA algorithms in a wide variety of scenarios, including interference and mobility. We fix transmit power to $11dBm$ and enable packet aggregation. We measure UDP throughput and PER, and average the results over 5 runs. The floorplan of our semi-open office, experimental environment is shown in Fig. 6.13, where the letters represent node locations. We note that this evaluation is based on a reduced set of 11 nodes. This reduced set is carefully chosen to include links with different characteristics (LoS, non-LoS, and a wide range of received signal strengths).

In our implementation of ARAMIS, we faced restrictions where the available chipset code does not support enabling an HTC field for 802.11n feedback. We mitigate this issue by implementing netlink sockets and transmitting packets with the HTC field over the wire from the receiver to the transmitter driver code. As a result, we believe the performance in our evaluation is a lower bound. Although transmitting feedback over the wired ensures no delivery loss, note that if a packet is not successfully received (e.g. there is a collision or loss), feedback will not be generated over the wire, as no ACK will be sent over the air. If the packet is

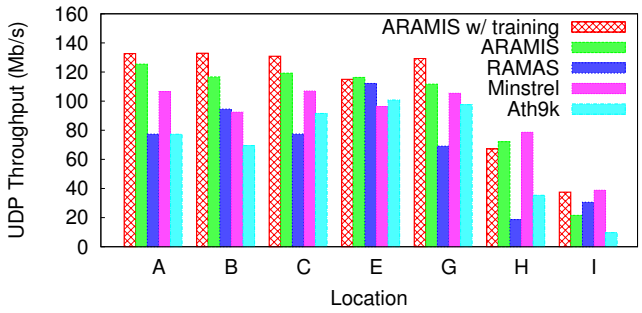


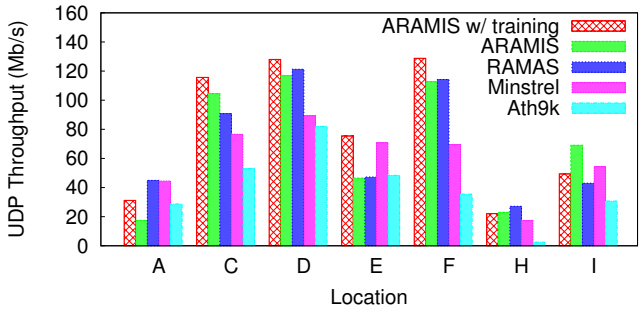
Figure 6.14: Algorithm performance in an interference-free environment.

successfully received, the loss of an ACK is unlikely as shorter ACK frames are sent at low, reliable rates, while the feedback over the wire is always received with a larger delay. The overhead of user-space-kernel communications, though minimal, often lead to delayed rate feedback receptions that trigger timeouts that mimic ACK packet losses.

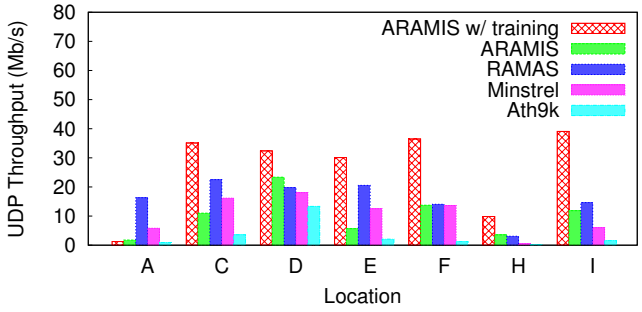
Moreover, the devices do not provide open access to the hardware generated Block-ACK at the receiver. This leads to inaccurate PER measurements, which reduces the precision of the ARAMIS training mechanism, explained in Section 6.5.3, and the accuracy of measured $E_k^{m,B}$ samples.

Testbed Results

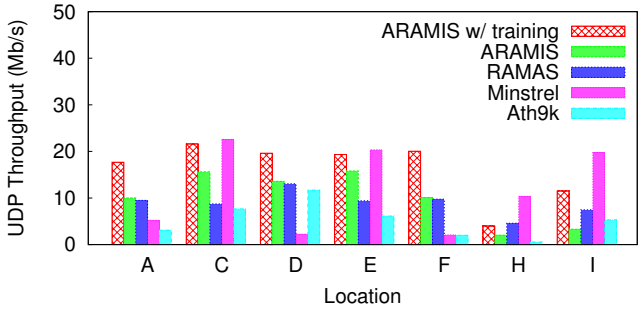
Fig. 6.14 and 6.15 show that ARAMIS consistently outperforms other algorithms in all test cases, with an average of 0.62 fold and up to a 2 fold throughput



(a) Adjacent 40MHz interferer.



(b) Adjacent 20MHz interferer.



(c) Channel sharing with a 20MHz interferer.

Figure 6.15: Algorithm performance under interference conditions.

increase in interference-free environments, an average of 2 fold and up to a 10 fold increase in interference conditions, and a 25% increase in mobile environments.

Interference-free: To assess how well each algorithm handles random channel loss, for example due to shadowing or multipath, Fig. 6.14 shows the performance in an interference-free environment at seven different locations. Even without the training mechanism, ARAMIS outperforms other algorithms with throughput gains of up to 26% and an average of 16% over Minstrel HT, up to 124% and an average of 90% over Ath9k, and up to 287% and an average of 79% over RAMAS. Note that our results for RAMAS are somewhat different from those reported [76], since they were obtained in different scenarios. RAMAS was previously evaluated only on the 2.4GHz range, which significantly limits the performance benefits of 802.11n features [83, 99].

RAMAS leads to an average PER of 11% and a maximum of 20%. The credit scheme it uses to adapt the number of streams is conservative, while the scheme to adapt the modulation and coding is aggressive. This mismatch causes RAMAS to often operate at sub-optimal rates with high modulations and single stream (e.g. MCS 7), which leads to high PER and reduced performance. Ath9k and Minstrel HT's random sampling incurs high overhead that results in poor performance. Ath9k also assumes PER monotonically increases with rate, which causes it to often ignore suitable high rates.

ARAMIS relies on our link predictor for rate selection and hence does not require random sampling. Its link prediction accuracy and ability to adapt MCS and bandwidth on a per-packet basis maximize opportunities to exploit more aggressive rates without sacrificing PER. We observe an average PER between 4 and 6%. ARAMIS is therefore suitable for low error tolerance applications, such as online gaming and bulk file transfers.

Interference: We now assess how the algorithms perform under interference from signal leakage, hidden nodes, and channel sharing.

Signal leakage is produced by transmissions on adjacent channels and can result in collisions similar to the hidden node problem. We evaluate how the algorithms react to interference due to leakage with varying interferer bandwidth, as we discovered that the impact of leakage varies according to channel width [20]. Fig. 6.15(a) presents results with an interfering link that operates on an adjacent 40MHz channel, while Fig. 6.15(b) for an adjacent 20MHz interferer.

Ath9k and Minstrel HT respond frequently and rapidly to interference by reducing the rate. Reducing the rate exacerbates the impact of leakage; frame transmission time increases and so do the opportunities for collisions. Similarly, RAMAS responds to channel disturbances by first reducing the number of streams, thus reducing the transmission rate.

With signal leakage, the reported SNR may be low and collisions could be interpreted as wireless losses. ARAMIS's PRR predictions hence may not match the measured values from the training mechanism. When the prediction error $E_k^{m,B}$ exceeds a given threshold, which we set to 0.2 based on our experiments, ARAMIS interprets that there is a collision problem and limits the influence of the training mechanism; it sets $E_k^{m,B}$ to the maximum allowed value, thus maintains transmissions at suitable high rates. For an adjacent 40MHz interferer shown in Fig. 6.15(a), we improve the throughput by an average of 10% and up to 60% over RAMAS, an average of 25% and up to 85% over Minstrel HT, and an average of 192% and up to 782% over Ath9k. For an adjacent 20MHz interferer shown in Fig. 6.15(b), the improvement is an average of 88% and up to 220% over RAMAS, an average of 400% and up to 412% over Minstrel HT, and an average of 1900% and up to 1908% over Ath9k. We observe greater performance improvements with an adjacent 20MHz interferer, since it is the more harmful configuration [20], and ARAMIS mitigates this interference.

Although the cap on the error threshold mitigates the effect of interference on RA, it can also degrade performance, as seen from location A in Fig. 6.15(b): ARAMIS first selects a rate which it identifies is best under interference-free conditions, and then reacts to high losses by capping the error threshold to 0.2. We find that, in cases such as location A when a strong link suddenly becomes extremely

lossy due to strong interference from channel leakage (or other non-802.11 interference), the error threshold forces ARAMIS to stay at higher rates than best, leading to performance losses.

We also investigate the channel sharing scenario with an interferer on a 20MHz channel. This scenario has been shown to create worse fairness issues than a 40MHz co-channel interferer whereby the slower 20MHz channel occupies the medium for longer periods of time [20]. In Fig. 6.15(c), we evaluate how well the algorithms perform under such conditions.

The presence of co-channel interference slightly increases collision probability, and thus $E_k^{m,B}$ increases, but remains under its maximum allowed value. As a result, the probability of using high rates is slightly reduced and Minstrel HT matches ARAMIS's performance in some locations since those collisions seldom affect Minstrel's random probing mechanism. At locations H and I in Fig. 6.15(c), we notice that channel sharing coupled with poor channel conditions can hamper the performance of ARAMIS. Channel sharing limits the number of opportunities to transmit, and if channel conditions are already poor and most transmissions are lost, this phenomenon can trigger ARAMIS's expiration timer, leading to frequent fall-backs to slow MCS. This phenomenon motivates the need for a dynamic expiration timer based on channel conditions.

The timely detection and adaptation to the channel conditions give ARAMIS an advantage over other algorithms, and this advantage is also evident in channel sharing conditions. At all locations, ARAMIS maintains the high order rates, thus exploiting its available channel time. ARAMIS improves the throughput by up to 76% over RAMAS, 251% over Minstrel HT, and 366% over Ath9k.

Mobility: We create a mobility scenario to evaluate the responsiveness of ARAMIS to rapidly changing channel conditions. With a static AP placed at Location I, we move the client on a trolley through the adjacent corridor from the indicated P_1 to P_2 at an approximate speed of 5km/h. ARAMIS achieves throughput of 80.27Mb/s and improves the throughput by 25% over RAMAS, 7% over Minstrel HT, and 15% over Ath9k. The small differences in throughput in this case may be due to the fact that ARAMIS is close to the capacity of this channel, which is low.

Note that our ARAMIS implementation had to overcome significant limitations due to hardware restrictions. These limitations reduce the potential performance benefits of ARAMIS. Hence, we believe that the ARAMIS performance we observe from our experiment is a lower bound.

6.7 Related Work

Wireless Link Metrics: A significant body of work has proposed methods to characterize link performance. RSSI, which is the most accessible link metric, has traditionally been used to identify a link's maximum expected throughput. Recent studies [3, 40, 93] have shown that RSSI is an unreliable metric to accurately predict performance. The utilization of *effective SNR* [40] is proposed, where the metric is generated using CSI feedback to accurately reflect link conditions in OFDM environments. However, complete CSI information could be costly to obtain and store [18] and is therefore not supported by all 802.11n devices.

Rate Adaptation: Rate adaptation has been one of the most popular research topics in WLANs [13, 109, 113], and new algorithms for 802.11n networks have been proposed [**?**, 40, 62, 83, 115]. Although solutions for legacy clients have been effective, they fall short when applied in 802.11n OFDM-MIMO settings [83]. Existing 802.11n solutions require either costly CSI [40, 86] or some form of a guided search (e.g., by probing candidate rates) to determine the best operating rate [83], which is inefficient when the search space is large. Other algorithms for MIMO environments do not consider other 802.11n features, such as channel bonding [**?**, 53], or consider alternative energy efficiency goals [62].

6.8 Conclusion and Future Work

The 802.11n standard has been touted as a new revolution in Wi-Fi technology, in part because of the number of new mechanisms that enable a multifold increase in transmission speeds relative to 802.11a/b/g. What is clear, however, is that while 802.11n has the theoretical ability to attain wireless data rates as high as a few hundred Mbps, it is only through intelligent and adaptive transmission strategies that such throughputs have a hope of being achieved. Among the most crucial questions for accessing the medium is the mechanism to select an appropriate data rate and bandwidth combination for transmission that is correctly responsive to changes in signal quality.

We have introduced ARAMIS, a closed-loop RA solution that jointly adapts rate and bandwidth. Through ARAMIS, we have *developed a medium access technique, and particularly an RA solution, that exploits 802.11 next-generation, MIMO high-speed links*. Given the high costs of adopting CSI in 802.11n environments, ARAMIS identifies and adopts a new method to quantify performance in 802.11n MIMO environments, and through trace-driven simulations, we have shown that ARAMIS closely approximates an optimal solution.

Through further implementation of our solution, we have demonstrated that ARAMIS obtains impressive performance gains over leading 802.11n rate adap-

tation contenders, including up to a 10 fold increase in throughput. We believe that ARAMIS is a critical component of a fully adaptive, intelligent 802.11n management system that dynamically optimizes 802.11n performance in response to changing channel conditions commonly present in operational wireless networks. Our solution design can also be applied in the context of the emerging 802.11ac standard, where MCS and channel width selection are faced with further challenges. Finally for future work, it would be useful to evaluate the benefits of ARAMIS's error threshold on the performance of TCP-based applications.

Chapter 7

Rate Adaptation for OFDM MU-MIMO 802.11ac WLANs

7.1 Introduction

In Chapters 5 and 6, we study, develop, and propose medium access techniques to exploit the exponential data rates enabled through next-generation, high-bandwidth WiFi technology, namely through wider channel widths and MIMO smart-antenna technology in 802.11n. In this chapter, we maintain our efforts towards enabling the next-generation, high-bandwidth links supported by the added IEEE 802.11 WiFi technologies, as the evolution of WiFi data rates continues to play a critical role in sustaining the exponentially growing demand for wireless network capacity. In this chapter, we investigate a key new technology in 802.11ac: Multiuser (MU) MIMO. The 802.11ac standard promises peak data-rates exceeding gigabit speeds, and MU-MIMO is a key new technology in 802.11ac that can enable such high data rates.

Until the emergence of the IEEE 802.11ac standard, WLAN link operation, e.g. for 802.11n, constituted of a single communicating transmitter and receiver pair equipped with multiple antennas to enable SU-MIMO communication. With the introduction of MU-MIMO in 802.11ac, the typical WLAN link model of a single communicating transmitter and receiver pair has been expanded such that an AP can now also communicate simultaneously with multiple clients via a transmission technique referred to as *downlink MU-MIMO* [2]. For a downlink MIMO transmission, at the beginning of a transmission opportunity, the AP selects multiple clients among the active clients based on their downlink traffic demands and the channel state information, determines appropriate signal encoding to minimize interference due to simultaneous transmission to multiple clients, and then transmits to the selected clients.

Downlink MU-MIMO introduces new system challenges across the different layers of the WLAN stack: these include the physical layer signal processing techniques at the transmitter and the receiver(s), the channel feedback mechanism, and medium access control and resource allocation issues such as deciding between MU-MIMO and SU-MIMO at any transmission opportunity, selection of clients for MU-MIMO transmission, and data-rate assignment to each client receiving MU-MIMO data stream(s).

In order to serve multiple clients simultaneously, prior work in communication theory literature has proposed multiple coding techniques with varying performance bounds. The coding scheme that has been shown to achieve full downlink MU-MIMO capacity is called Dirty Paper Coding [17], and is based on pre-subtracting interference from transmission intended for one receiver to the other MU-MIMO receivers. However, DPC is computationally prohibitive in practical systems. As a result, a number of alternative, suboptimal strategies with reduced complexity have been proposed [100, 122, 123]. These approaches either precode signals intended for different clients so as to avoid any interference among them [100, 122, 123] or employ other means to reduce the interference leakage from a transmission destined for one client to the other [94]. For example, [123] employs Zero-Force Beamforming (ZFBF) and provides an elegant user selection scheme for MU-MIMO systems with single-antenna clients assuming a flat fading channel. For a similar narrowband setting, [100] proposes Block Diagonalization that extends the ZFBF concept to multi-antenna clients and provides more capacity than ZFBF by exploiting cooperation among receive antennas belonging to the same client [100].

Unfortunately, many assumptions in prior work do not hold for 802.11ac environments where clients, equipped with multiple antennas, can support both SU-MIMO and MU-MIMO, and are expected to operate efficiently over frequency selective channels using Orthogonal Frequency-Division Multiplexing (OFDM, a ro-

bust multicarrier communication technique). Besides, the 802.11ac standard also enforces other constraints such as use of the same modulation and coding scheme (MCS) for all the streams to any single client. Our main contribution in this work is to identify some key concepts towards the application of MU-MIMO to 802.11 environments so as to exploit its benefits in the best possible way.

We define a foundational 802.11ac WLAN system model and identify three important issues in the physical layer: practical MU-MIMO transmit precoding schemes, receive processing at the MU-MIMO clients, and efficient channel feedback methods. Referring to the state-of-the-art in the existing communication theory literature and accounting for the unique WLAN OFDM-MIMO systems constraints outlined earlier, we propose a framework that employs block diagonalization [100] for transmit precoding and the corresponding receive processing at the clients.

An effective channel feedback mechanism is crucial in order to facilitate 802.11ac MU-MIMO client selection, transmit precoding and receive processing. MU-MIMO channel feedback has been a topic of intense interest over the last few years given its role in enabling practical MU-MIMO communication [12, 90–92]. In this work, we refer to proven concepts from related literature in order to support and corroborate our design choices. Particularly, we use higher resolution channel feedback from clients chosen for MU-MIMO transmission, as prior work shows the need for a higher resolution channel feedback for ensuring MU-MIMO capacity gains [92]).

In addition to investigating the transmit-receive signal processing, our work identifies and addresses two critical medium access and resource allocation problems for a practical MU-MIMO 802.11 system: transmit mode (SU-MIMO versus MU-MIMO) selection and data-rate selection for the chosen user(s) in the transmission set. The transmit mode selection as well as the data rate selection utilize the detailed channel feedback and use a utility metric maximization approach.

We rely on extensive simulations in order to assess the efficacy of our system solution on the application of MU-MIMO in 802.11 multi-carrier OFDM WLANs. In particular, we evaluate the efficacy of our MU-MIMO signal processing method on the performance of a MU-MIMO system versus a traditional SU-MIMO system for varying user locations and user diversity, for varying and widely-used wireless channel models in literature [32, 75, 104]. We find that the performance benefits of our system solution consistently performs better than the best SU-MIMO approach, thus achieving the promise for higher bandwidth with MU-MIMO in 802.11ac. Further, in the cases where MU-MIMO degrades performance, transmit mode selection would switch back to a SU-MIMO transmission, thus adding a critical component to our solution design. Finally, our evaluations motivate the need for intelligent MU client selection approaches, which we consider as future work to complement our existing system design.

The rest of the chapter is organized as follows. Note that in the context of 802.11ac WLANs, we use MU-MIMO and downlink MU-MIMO interchangeably in this chapter since uplink MU-MIMO is not part of the 802.11ac specifications. Section 7.2 outlines some challenges and goals towards enabling MU-MIMO in WLANs, also accounting for constraints imposed by the 802.11ac standard. We discuss our downlink spatial multiplexing scheme in Section 7.3, then propose a practical RS algorithm 7.5, which accounts for the impact of the coding scheme on the channel properties between the client and the AP. We evaluate components of our solution in Section 7.6.

7.2 Design Considerations

To identify the appropriate MU-MIMO signal processing technique as well as the ensuing and final rate selection process, we are influenced by the inherent characteristics of an 802.11 OFDM multi-carrier enrvionment, the requirements set by the 802.11ac standard, and the goals we have on performance and practicality.

7.2.1 Design Challenges and Goals

To guide the design of our solution, we first identify some key challenges towards enabling practical MU-MIMO in 802.11ac WLANs. We then describe our corresponding system design choices to address these problems.

Intelligent Channel Sate Information (CSI) feedback: Since linear MU-MIMO precoding techniques such as block diagonalization (BD) and zero forced beamforming (ZFBF) rely on canceling the interference from the transmission to one client on other simultaneously receiving clients that are part of the same *MU-MIMO receiver group*, the spatial relationships among the client channels determines the maximum spectral efficiency that can be provided by such approaches. The CSI of the intended receivers is therefore essential to derive appropriate MU-MIMO precoding matrices to cancel the interference via BD or ZFBF. Prior work has investigated the relationship between channel feedback accuracy and resolution versus the MU-MIMO capacity gains for popular MU-MIMO precoding techniques such as BD and ZFBF [12, 90–92]. For instance, [92] highlights the sensitivity of MU-MIMO capacity gains to the resolution or accuracy of the channel feedback used for calculation of precoding matrices. This work shows that for a fixed channel feedback overhead budget, it is advisable to get higher resolution feedback (even if for a subset of clients, rather than obtaining a low resolution feedback from all the clients

for the purpose of calculating the precoding matrices). This is because inaccurate feedback can lead to"leaked interference" between MU-MIMO clients and lead to poor capacity. Communicating CSI in real networks however incurs high overheads and can quickly become costly in large network environments [19]. At the same time, since the scope of our work is concerned with the final and limited set of users within a given 802.11ac MU-MIMO transmission opportunity, we refer back to high resolution CSI from the limited set of clients to enable higher MU-MIMO performance.[1]

Transmit and receive signal processing: Central to the implementation of an effective downlink MU-MIMO solution of WLANs is identifying the appropriate signal-processing mechanisms at both the transmitter (AP in downlink) through *precoding*, and the receivers (clients in downlink) through *receiver shaping and MU-MIMO equalization*. The performance (in terms of aggregate capacity), practicality (robustness to different deployment environments as well as channel feedback requirements relative to other schemes) and the complexity (computation and memory overhead at both the AP as well as the clients) of these schemes are key metrics that guide our choice of appropriate signal processing schemes. We then also incorporate 802.11ac specific constraints described later in this section in order to refine these schemes for better performance in tune with the 802.11ac deployments.

[1]IEEE 802.11ac allows only up to four clients to associate with an AP in a MU-MIMO transmission.

AP transmit mode selection: While MU-MIMO *can* provide significant performance benefits over SU-MIMO through the transmission of multiple signals to different clients simultaneously, exploiting the inherent spatial diversity due to their separation (unlike SU-MIMO that relies on multiple antennas located at the same client that are not separated widely for small form factors and can see correlated channels or limited spatial diversity in many 802.11ac deployment settings) there is no guarantee that this will always be the case. The success of a MU-MIMO transmission depends on the orthogonality space between participating clients. By forcing each client into the orthogonality space of other clients in the transmission set, the capacity performance of each individual client is reduced in the MU scenario compared to the SU scenario [96]. In some cases, that reduction can be detrimental to the overall performance. *MU-MIMO systems must therefore adopt intelligent methods of evaluating the alternative SU-MIMO performance and switching back if appropriate.*

The post-coding effective channel and rate selection: The final step of applying MU-MIMO to 802.11 WLANs is data rate (equivalently, the modulation and coding scheme) selection for each chosen client. After executing the necessary steps to identify the clients and their respective MIMO channel eigenmodes for MU-MIMO decoding and applying the appropriate signal processing, we now assign data rates or a *single MCS* to all MU-MIMO data streams for each client while the chosen MCS

for one client can be different from the other. It is important to *identify the effective channel for each client as a result of MU-MIMO operation to perform accurate rate selection.*

7.2.2 IEEE 802.11ac Standard Constraints

In addition to the design goals highlighted in the preceding section that apply to MU-MIMO application over any wireless system, we identify two main challenges towards the application of MU-MIMO specifically to 802.11ac WLANs.

OFDM multi-carrier environment: Any MU-MIMO proposal for 802.11 WLANs must address the frequency-selective, multi-carrier nature of 802.11 OFDM environments [40]. Most of the prior proposals for MU-MIMO are designed with an assumption of flat-fading, single-carrier channels. A direct application of existing solutions to multi-carrier environments will introduce high computational overhead that would render existing solutions infeasible for practical deployments. Recent work quantifies this overhead and demonstrates that more context-specific thought is required in the selection of a signal processing technique for MU-MIMO OFDM environments [97].

Single MCS per MIMO client: The 802.11ac specification mandates applying the same MCS across all streams of a single MIMO client. This requirement constrains

the design of MU-MIMO techniques to achieve high spectral efficiency. In a more straightforward scenario, the MCS will be catered to each stream separately, and this eliminates the impact of one stream on the performance of the other through exclusive coding. Establishing a common MCS among the multiple streams of a single client requires careful design to compensate for the likely impact of the poor stream/s on the strong stream/s and the resulting performance.

Our work proposes a MU-MIMO approach that achieves our design goals while addressing the challenges outlined above. We next present an overview of our solution, summarizing the signal processing and rate selection components, followed by a detailed discussion of their functionality.

7.3 Overview

In this work, we propose methods towards the application of MU-MIMO in 802.11 WLANs. We consider a model, as shown in Figure 7.1, with K clients associated with a single AP. The AP is equipped with T transmit antennas and the kth client with N_k receive antennas. We define $l_k \leq N_k$ as the number of distinct MIMO streams assigned to client k. The signal received by a participating client k may be written as:

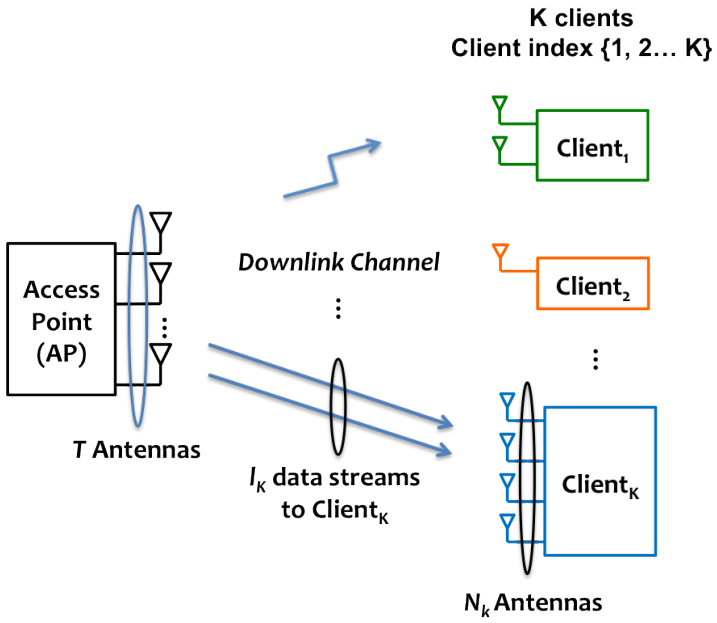


Figure 7.1: Diagram of MIMO downlink system with T transmit antennas at the AP and K clients. Each client $k \in K$ has N_k receive antennas, and each participating client receives l_k data streams from the AP.

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{M}\mathbf{x} + \mathbf{n}_k = \mathbf{H}_k\mathbf{M}_k\mathbf{x}_k + \mathbf{H}_k \sum_{i=1, i\neq k}^{K} \mathbf{M}_i\mathbf{x}_i + \mathbf{n}_k \tag{7.1}$$

where $\mathbf{x}_k \in \mathbb{C}^{l_k \times 1}$ is the signal transmitted to client k, $\mathbf{M}_k \in \mathbb{C}^{T \times l_k}$ is the precoding matrix, $\mathbf{H}_k \in \mathbb{C}^{N_k \times T}$ is the channel gain matrix to k, $\mathbf{n}_k \in \mathbb{C}^{N_k \times 1}$ is the noise and $\mathbf{y}_k \in \mathbb{C}^{l_k \times 1}$ is the received signal at client k. Furthermore, since we are dealing with a multi-carrier environment, a separate Eq. 7.1 is used for each OFDM subcarrier $s \in S$, for each user k, whereby $\mathbf{H}_{k,s}$ represents the channel at subcarrier s for client k.[2]

Our solution process starts at each transmission opportunity during a Controlled Access Period (CAP) where the AP has information about the clients for which it has non-empty downlink traffic and wants to transmit to a select subset of these clients in a controlled HCCA mechanism [2]. The channel information is then collected for this subset of clients with a higher resolution.

With the higher resolution MU-MIMO receiver group clients' CSI in place, we can precode the signals transmitted to each client in a MU-MIMO environment using *transmit precoding* where we use a linear precoding technique called block diagonalization that has been shown to be simple and effective [100]. Our transmit precoding approach relies on a *receiver shaping* mechanism performed at the client post signal reception, where each client's MIMO antennas coordinate to maximize gain. Then,

[2]We represent matrices with uppercase boldface letters and vectors with lowercase boldface letters.

by evaluating the performance of MU transmission compared to the best SU-MIMO transmission, we perform *transmit mode selection* at the AP between MU-MIMO and SU-MIMO. If MU-MIMO mode has been selected, the receiver shaping matrix will then be fed back to the client in standard-compliant receiver shaping packets [2]. Finally, *rate selection* is performed on the effective channel to each client, subject to the single MCS per client 802.11ac constraint.

In the following sections, we describe in detail the precoding and rate selection components of our solution design, addressing the challenges discussed earlier and describing the insights we gained towards MU-MIMO operation.

7.4 Downlink Spatial Multiplexing in OFDM MU-MIMO

A design choice of our signal processing algorithm is to support multiple data subchannels per user with low computational cost. Allowing multiple data subchannels means that we define $l_k \leq N_k$ for $N_k > 1$. This implicitly assumes coordination between the multiple receive antennas at the client to successfully separate each distinct l_k signal, which is accomplished through *receiver shaping*. This is a reasonable assumption to make as this reflects current implementations of 802.11 MIMO clients [39]. The receiver shaping applied however is first influenced by the

precoding we apply at the AP, pre-transmission. Precoding is a signal processing method that enables MU-MIMO transmission at the AP.

We now discuss the precoding and receiver shaping mechanisms, which are based on the *Block Diagonalization* (BD) approach we adopt from related literature. BD is an extension of the popular Zero-Force Beamforming (ZFBF) [122] concept for MIMO (vs SISO) clients, and it can achieve higher gains than ZFBF by exploiting full client cooperation through receiver shaping [100]. In fact, using the BD receiver shaping mechanism, we can maximize channel gain.

Precoding in a multi-carrier environment is faced with the challenge of having to deal with the cost associated with performing computations on a full CSI. However, due to related work that emphasizes the dependency of MU-MIMO performance on high resolution CSI [92], we return to a higher-resolution version of $\mathbf{H}_k$. High resolution channel information allows us to capture more fine-grained frequency-selective fluctuations or variations across the OFDM subcarriers. We perform precoding and receiver shaping on a high resolution channel per client

7.4.1 Block Diagonalization

Precoding and Receiver Shaping: We now outline the approach to compute the precoding matrix M_k from Equation 7.1. To do so, each client that is participating in a transmission opportunity would project its signal or channel to the direction

orthogonal to the other clients in the transmission set. This ensures that client transmissions do not interfere with each other. The process of projecting a client j's signal onto the null space of other clients starts by identifying the channels that are occupied by all other clients, excluding client j, which is given by:

$$\tilde{\mathbf{H}}_j = [\mathbf{H}_1^T ... \mathbf{H}_{j-1}^T \mathbf{H}_{j+1}^T ... \mathbf{H}_K^T]^T \tag{7.2}$$

The null space of the users in the transmission set, excluding user j, is then computed using the SVD of $\tilde{\mathbf{H}}_j$, such that:

$$\tilde{\mathbf{H}}_j = \tilde{\mathbf{U}}_j \tilde{\mathbf{\Sigma}}_j [\tilde{\mathbf{V}}_j^{(1)} \tilde{\mathbf{V}}_j^{(0)}]^* \tag{7.3}$$

The null space of $\tilde{\mathbf{H}}_j$ is $\tilde{\mathbf{V}}_j^{(0)}$, the right singular vectors corresponding to the vanishing zeros in $\tilde{\mathbf{\Sigma}}_j$, i.e. the paths that are not occupied by clients in the current transmission set, excluding user j. As a result, the effective SU-MIMO channel of j in this MU-MIMO environment is $\mathbf{H}_j \tilde{\mathbf{V}}_j^{(0)}$, which is the effective channel remaining to user j from participating in a MU transmission. BD is a process that essentially converts a MU-MIMO channel into multiple SU-MIMO channels. Principles that are applied to a typical SU-MIMO setting can now be applied to the effective channel of each individual MIMO client.

In order to maximize the information rate at a client j, the effective channel of j can be further beamformed along its dominant eigen values or streams, while subject

to maintaining zero interference to other clients. In a MU-MIMO transmission, we consider the dominant eigen values of each user's effective channel. We compute a second SVD from each client j's effective channel, such that:

$$\mathbf{H}_j \tilde{\mathbf{V}}_j^{(0)} = \mathbf{U}_j \mathbf{\Sigma}_j [\mathbf{V}_j^{(1)} \mathbf{V}_j^{(0)}]^* \tag{7.4}$$

Where the dominant eigen values are defined by $\mathbf{V}_j^{(1)}$. The final modulation applied to user j is therefore:

$$\mathbf{M}_j = \tilde{\mathbf{V}}_j^{(0)} \mathbf{V}_j^{(1)} \tag{7.5}$$

By assuming coordination between the receiving antennas, we can maximize channel gain. This is done by predicting the optimal receiver. The optimal receiver will decode the received signals over the dominant eigen modes or streams from the effective channel between the transmitter and receiver pair. Since the effective channel is determined post-BD, as shown in Equation 7.4, the optimal receiver can only be predicted or guessed pre-BD. Therefore, assuming that a client j will receive m_j streams from the AP, the receiver shaping or weighting matrix $\mathbf{W}_j$ is the first m_j columns of the left singular vectors of the SVD of $\mathbf{H}_j$.

Transmit power distribution: Waterfilling is a popular approach used in literature, and it can be used for 802.11ac, however it leads to high complexity [32]. Simpler methods of distributing power have been proposed in literature, and they have shown

that the equal transmit power distribution approach to each client achieves higher receive power and sum rates than other simple approaches in 802.11ac MU-MIMO multipath fading channels [30]. We build on this approach and consider equal power allocation across all streams allocated to all clients, rather than for each client alone, during the stream selection process.

Receiver shaping feedback from AP to client: In order to effectively decode the precoded symbols, the AP will have to feedback the receiver shaping matrix or vector to the client. In the case of precoding using Block Diagonalization, the receiver shaping matrix that cancels the precoding performed at the AP and allows for the successful reception of the transmitted signals at their dedicated client is given by $\mathbf{W}_j\mathbf{V}_j^{(1)}$

7.4.2 AP Transmit Mode Selection

In IEEE 802.11ac networks, where performance from MU-MIMO is not guaranteed to outperform SU-MIMO in all cases, we finalize our implementation by evaluating and selecting the transmit mode that achieves the highest capacity. We now outline how we compute capacity in each case, where we define $\mathbb{P}$ as the maximum allowable transmit power [2].

MU-MIMO Capacity: In order to compute the capacity for each client in the MU-MIMO case, we must compute the effective SU-MIMO channel of that client [32]. Due to BD precoding, the effective SU-MIMO channel of each client in a MU-MIMO transmission is defined as shown in Equation 7.4.

$$R_{\mathbb{S}} = \sum_{i=1}^{rank(\mathbf{H}_j \tilde{\mathbf{V}}_j^{(0)})} \log_2(1 + (\lambda_{j,i}^2 P_i/\sigma_n^2) \tag{7.6}$$

Where $P_i = \mathbb{P}/(|\mathbb{S}| \times Sb)$ is the power in dBm per stream i, $\mathbb{S}$ is the number of clients in the transmission opportunity, and $Sb = rank(\mathbf{H}_j \tilde{\mathbf{V}}_j^{(0)})$ is the number of independent streams for client j.

SU-MIMO Capacity: We refer to the user selected for SU-MIMO transmission as π. This user might support only a single stream, in which case we compute the diversity capacity, or it could support more than one parallel data channel, in which case we compute the multiplexing capacity. The SU-MIMO multiplexing equation is given by, where $P_i = \mathbb{P}/(rank(\tilde{\mathbf{H}}_\pi) \times Sb)$ is the power in dBm per stream:

$$R_{\mathbb{S}} = \sum_{c=1}^{Sb} \left(\sum_{i=1}^{rank(\mathbf{H}_\pi)} \log_2(1 + (\lambda + \pi, i^2 P_i/\sigma_n^2) \right) \tag{7.7}$$

As for the SU and single-stream diversity capacity, we compute it as:

$$R_\pi = \log_2(1 + \mathbf{H}\mathbf{H}^*) \times 10^{(P - PndBm)/10} \tag{7.8}$$

Where $P = \mathbb{P}/(|\mathbb{S}| \times Sb)$.

7.5 Rate Selection

Rate Selection (RS) in 802.11 WLANs is the process by which the transmitter intelligently determines the best transmission rate to its receiver(s). An effective RS solution achieves this by mapping the channel properties over which the wireless signals are sent to the corresponding highest transmission rate that satisfies the bit error rate requirements.

Due to MU-MIMO precoding, the effective channel for each client is a function of the precoding matrix and the real channel. Since RA relies on accurate channel properties to determine the best rate, it should now account for the impact of precoding on the channel to each client.

BD cancels interference from the simultaneously transmitted signals destined to other participating clients. This means that from Equation 7.1, $\mathbf{H}_k \sum_{i=1,i\neq k}^{K} \mathbf{M}_i \mathbf{x}_i = 0$ and the broadcast channel is decomposed into parallel SU-MIMO channels $\mathbf{y}_k = \mathbf{H}_k \mathbf{M}_k \mathbf{x}_k + \mathbf{n}_k$ for each participating user k. The implication of BD is that the RS will depend on the properties of $\mathbf{H}_k \mathbf{M}_k$ to compute the rate for client k.

We use $\mathbf{H}_k \mathbf{M}_k = \mathbf{H}_{k,eff}$ to represent the effective channel accounting for BD precoding for a client k. It is given by $\mathbf{H}_{k,eff} = \mathbf{H}_k \tilde{\mathbf{V}}_k^{(0)}$, where $\tilde{\mathbf{V}}_k^{(0)} (= \mathbf{M}_k)$ is the precoding vector that represents the null space of the channels of all participating clients excluding client k. The BD precoding vector $\tilde{\mathbf{V}}_k^{(0)}$ thus eliminates the inter-

ference from transmissions to other clients to client $k \in K$ and allows all participating clients (such client sets are assumed to be pre-determined here) to successfully decode their corresponding signals.

Computing $\mathbf{H}_{k,eff}$ decomposes the RS problem for MU-MIMO to the corresponding SU-MIMO RS problem for each participating client k, by identifying the properties of the channel to k, post-precoding. This conversion allows us to treat the channel $\mathbf{H}_{k,eff}$ as a typical SU-MIMO channel.

Now that we have decomposed the MU-MIMO channel for a client k to the corresponding SU-MIMO channels via BD, we can apply SU-MIMO-OFDM RS techniques, as shown in Algorithm 4. We select the rate for each client k based on a popular link metric, referred to as *effectiveSNR* [40]. This metric uses subcarrier SNR information to predict the best rate for a MIMO-OFDM link. This technique computes the Bit Error Rate (BER) for each stream in a subcarrier and over all subcarriers, then finds the corresponding average BER for our effective SU-MIMO-OFDM link, and finally maps that average to SNR [32]. This SNR value is referred to as *effectiveSNR* and is mapped to the highest rate the link can support based on existing conversion models.

Algorithm 4 802.11ac MU-MIMO RS Algorithm

1: **for each** stream $\{n, k\} \in S$ **do**

2: $\mathbf{h}_{k,eff} = \mathbf{H}_k \mathbf{M}_k = \mathbf{H}_k \tilde{\mathbf{V}}_k^{(0)}$

3: **for each** rate $r \in R$ **do**

4: $\rho_{eff,k,r} = BER_m^{-1}\Big(\frac{1}{S}\frac{1}{l_k}\sum_{s\in S}\sum_{i\in l_k} BER_r(\rho_{s,i})\Big)$

5: **end for**

6: $MCS_{best,k} = \text{argmax}\left\{map_{r\in R}\Big(\rho_{eff,k,r} \rightarrow MCS\Big)\right\}$

7: **end for**

7.6 System Evaluation

We now evaluate the efficacy of our system approach. We particularly evaluate the relevance of a BD precoding and receiver shaping approach on the 802.11 OFDM performance, considering the constraints the 802.11 standard imposes on system parameters.[3] The purpose of our evaluation is two-fold: (1) to establish a firm confirmation that BD takes advantage of 802.11 channel conditions and constraints to gain multiuser spatial diversity gains in capacity, and (2) to gain an understanding of the impact of 802.11 on performance and what can be further improved on to maximize performance in 802.11 WLANs.

We implement our system in Matlab and evaluate the performance of our system solution for varying channel conditions. We select a set of channel models that

[3]802.11ac limits the number of streams per client to at most four streams, and the number of clients associate to an AP within a MU-MIMO transmission to four distinct clients.

recreate varying critical and fundamental wireless environments, namely: Line of Sight (LoS), Rician, and Rayleigh channel models [32, 75, 104]. The LoS model occurs when there is a direct, dominant line of sight path between a transmitter and receiver pair. The Rayleigh channel or fading model represents a non-LoS configuration and is a popular wireless propagation model used in literature that captures the fade in signal propagation due to the effect of heavily built-up urban environments on radio signals. Rayleigh is applicable when there is no dominant propagation path along a LoS between a transmitter and receiver pair, but if there is a dominant LoS component, a Rician fading model is used, which combines both LoS and non-LoS (i.e., Rayleigh component) paths between the transmitter and receiver. We believe these models are sufficient to capture behavior in 802.11 environments that would be useful for our evaluation, as described above.

We set the antenna separation at the AP and client to $\frac{\lambda}{2}$, where $\lambda = \frac{c}{f}$, where c is the speed of light and f is the frequency band. This antenna separation is typical in 802.11 WiFi devices and chipsets [39]. We also set the MIMO antenna configuration to 8×3, where $M = 8$ and $N_k = 3$ for each client $k \in K$. Within the context of 802.11 WLAN operation, and due to the random CSMA nature of channel access, we randomly select clients to participate within a MU transmission opportunity, which we refer to as *random MU selection*. Since we would like to gauge the performance benefits of a MU-MIMO approach, we compare it to an intelligent

SU-MIMO approach that selects the client that maximizes channel gain. This allows us to compare MU-MIMO with BD in an 802.11 WLAN with the best lower bound performance from a traditional 802.11 SU setup.

Our results are shown in Figures 7.3 and 7.2. In Figure 7.2, we start by establishing a controlled test case scenario to evaluate the performance benefits of applying MU-MIMO with BD in 802.11 WLANs, for varying channel models, and for varying client positions. We report performance as the normalized spectrum efficiency achieved using MU-MIMO with that achieved using a traditional SU-MIMO approach, where spectral efficiency is reported in Bps/Hz. Our controlled and limited test case scenario consists of two clients and an AP, where one client is at a fixed position in the 90° quadrant, and we shift the position of the second client over all points in the grid. Each sector or unit in the grid represents the normalized spectral efficiency when the second client is at that point in the grid. In the test cases shown, the first client is fixed at position $(\frac{X}{2}, \frac{Y}{2})$, however we find the behavior seen is representative and holds for varying locations of the fixed client.

From Figure 7.2, we can see that there is always a capacity gain with MU-MIMO, independent of the channel model. In fact, with BD, MU-MIMO becomes a superset of SU-MIMO at all locations, at varying gains. With MU, the AP can allow all streams from the first user, and it adds streams from other users onto the transmission opportunity, without incurring interference on the first user. As a result,

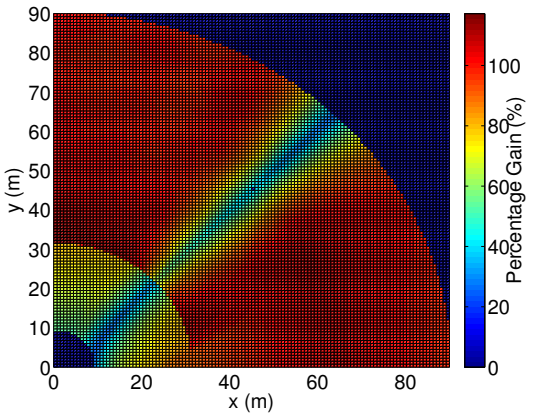
there will at least not be a loss in overall network gain. A pattern we see between all figures is the increase in MU spectral efficiency gain achieved through exploiting the inherent spatial diveristy achieved through the large spatial separation between the two participating clients with respect to the AP at location $(0, 0)$.

For example, in the Figure 7.2(a) for the LoS channel, since LoS is typically characterized by a single dominant line of sight path to the AP, placing the second client within close vicinity of the fixed client at $(\frac{X}{2}, \frac{Y}{2})$ would minimize the spatial diversity between those clients, thus minimizing the spectral efficiency gains from using MU-MIMO. This pattern is not clear in Figure 7.2(b), as a Rician channel exploits the random, non-LoS Rayleigh component to exploit spatial diversity, despite the proximity of the clients. The proximity will only affect the LoS component of the channel.

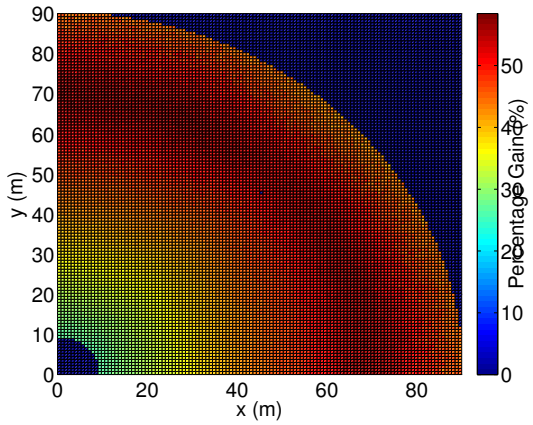
Another pattern that is clear is the lower gains of MU-MIMO compared to SU-MIMO when the second client is closer to the AP. In order to support MU, each participating client's channel with BD is projected onto the orthogonal space of other clients in the transmission set. This projection reduces channel gain. When a client exhibits high channel gain from proximity to the AP, SU will achieve high spectral efficiency from selecting that client that maximizes gain, and thus spectral efficiency. On the other hand, enabling MU-MIMO will reduce each of the two client's respective channel gain, thus minimizing the performance benefits.

Figure 7.2 has given us insights into the behavior of MU-MIMO at different locations compared to SU-MIMO. It is clear that MU-MIMO consistently increases the overall network spectrum efficiency, independent of the channel model. The Rician model in Figure 7.2(b) exhibits similar behavior to the Rayleigh model, which is an extreme Non-LoS version of the Rician model, and we therefore exclude the Rayleigh graph to avoid redundancy. At the same time however, the per-user spectral efficiency suffers. In such cases, our transmit mode selection approach would step in and switch back to the performance-maximizing SU-MIMO setting. Next, we evaluate the behavior of MU-MIMO in more general settings with a varying number of users.

Figure 7.3 shows the spectral efficiency in Bps/Hz with a varying number of users distributed in a 90° sector. Recall, that our MU approach is based on random selection, and we refer to it as *Random MU Selection*, while our SU approach establishes the highest lower bound performance for MU by selecting the single client that maximizes gain. It is clear that, independent of the channel model, the gain achieved through user diversity does not improve performance beyond a certain number of clients with sufficient user diversity. We now discuss the significant patterns in behavior that we see that leads to insights on MU-MIMO performance in 802.11 WLANs with BD.

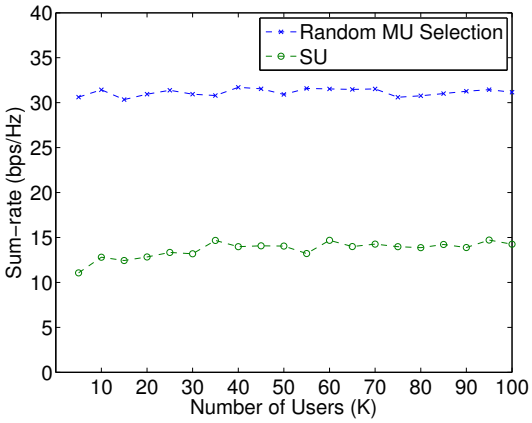


(a) Line of sight (LoS).

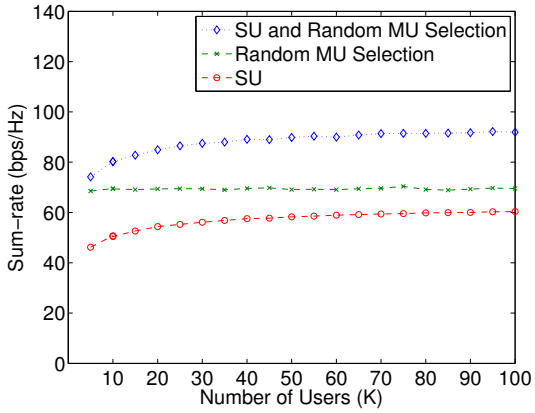


(b) Rician channel.

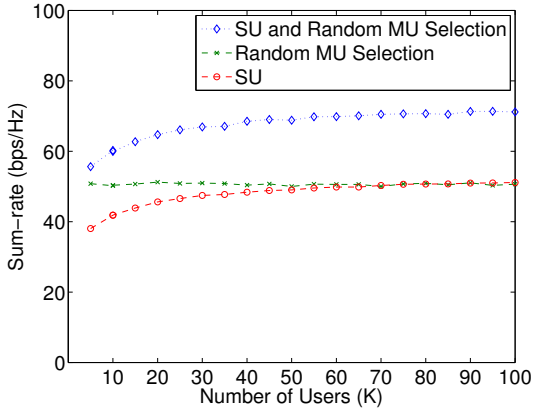
Figure 7.2: Normalized MU-MIMO sum rate capacity with respect to SU-MIMO capacity for two clients with varying channel models. One client's location is fixed in the middle of the grid at $(\frac{X}{2}, \frac{Y}{2})$, while the other client is evaluated at each other point in the grid, and the AP is at point (0,0).



(a) Line of sight (LoS).



(b) Rayleigh channel.



(c) Rician channel.

Figure 7.3: Sum rate capacity for a varying number of users with varying channel models.

We start by investigating performance for the LoS channel model, shown in Figure 7.3(a). By definition, a LoS SU-MIMO channel is characterized as low rank (i.e., few eigenmodes compared to the number of receive antennas, and primarily a single dominant eigenmode), particularly given the lower spatial diversity between closely spaced antennas at the AP and the clients in 802.11 devices and chipsets [39]. It is known that SU LoS MIMO only allows for effective multi-stream spatial multiplexing for specific separations between the antennas at both the AP and clients,[4] however 802.11 WiFi devices do not support such antenna separation. As a result, the SU-MIMO LoS capacity is more or less a single stream capacity for the chosen client, and the system does not benefit from multi-stream capacity gain.

Using random MU selection for LoS channel helps exploit the inherent spatial diversity that comes with the large spatial separation between the clients. Therefore, the rank of the overall MU-MIMO channel formed by the T transmit antennas at the AP and the total $R \times N_k$ receive antennas at the R selected clients with N_k antennas each, is typically $\min(T, R) = R$. As a result, many streams or effective spatial multiplexing can be supported between the AP and R clients, and given the typically good channel gain at each client (better than a fading channel with high probability) the overall MU-MIMO spatial multiplexing capacity turns out to be

[4]This separation is known as Rayleigh spacing, and it varies with the wavelength and the link distance and creates the spatial diversity leading to multiple eigenmodes with LoS.

high. This explains the significant performance gains of MU compared to SU in a LoS channel, as shown in Figure 7.3(a).

On the other hand, Rayleigh SU-MIMO channel is such that the rank is typically $\min(T, N_k) = N_k > 1$. This diversity is inherent even with relatively small separation (for ex. λ) which is enabled through multipath in the channel, leading to random values for the multiple eigenmodes. The rank of the MU-MIMO channel would be expected to be around T, assuming that $N_k \times R > T$. As a result, the rank of the MU-MIMO channel is $> N_k$ if $T > N_k$, but the characteristic spatial diversity or relationship between each eigenmode would be similar to that for SU-MIMO.

Since the SU approach in our case selects the client with the highest gain (among all the R participating clients), it gives an edge to the SU-MIMO spatial multiplexing capacity, leading to possibly N_k streams with high capacity. Since the R clients are picked randomly for the MU-MIMO case, they might form an T ranked MU-MIMO matrix, but the channel gains and therefore even the eigenmodes would typically be smaller, leading to less data carrying capacity, as compared with the SU-MIMO Rayleigh channel corresponding to the highest CQI user.

Therefore, in a Rayleigh channel, the rank of the MU channel T might be higher than the SU-MIMO channel, but the SU-MIMO channel's rank N_k is larger than 1 (where a rank of 1 is typical in LoS channels), and since our SU approach picks the client with the highest CQI, this means that those SU-MIMO channel eigenmodes

will on average be expected to have a high data carrying capacity than that for the typical eigenmodes for the higher ranked MU-MIMO formed with randomly picked users. As a result, the performance gain from MU-MIMO is reduced compared to the SU-MIMO approach, as can be seen in Figure 7.3(b).

The Rician channel model in Figure 7.3(c) exhibits the worst performance, as it combines the potential for low channel rank from its LoS component, and low gains from it non-LoS, Rayleigh fading component. In this case, the random MU-MIMO approach approximates the SU-MIMO approach, and there is no real channel gain. In cases where this scenario is replicated, our system would operate using SU-MIMO, as it is the performance-maximizing option, especially considering the reduction in per-user capacity that the MU-MIMO approach would lead to.

It is clear from Figure 7.3(b) and 7.3(c) that under certain common channel conditions, MU-MIMO could benefit from an intelligent client or user selection approach. We verify this by adding a slightly modified MU selection approach, where now the first client of the MU transmission set is the client selected from our SU approach. This means that the MU transmission set now includes the client that maximizes channel gain in its set of R clients. We refer to this approach in Figures 7.3(b) and 7.3(c) as *SU and Random MU Selection*. This approach is now a clear superset of our SU approach, and we regain much capacity gain from performing MU-MIMO on top of the SU approach.

Our evaluations in Figures 7.2 and 7.3 show that BD is an effective approach in 802.11 WLANs that is able to achieve the MU-MIMO channel gains. We motivate the need for transmit mode selection in scenarios where MU-MIMO impacts performance. Finally, we motivate the need for future work on intelligent client selection approaches. We believe that an effective client selection approach would identify the set of clients with maximum orthogonality between their channels, thus minimizing the loss in channel gain from projecting each client onto the orthogonal space of other participating clients. Further contributions would also consider proportional fairness, in terms of deprioritizing users with the highest average sum rate.

7.7 Conclusion and Future Work

In this chapter, we maintain our goal to *develop medium access techniques that exploit next-generation, high-bandwidth data rates enabled through added WiFi technology*. Specifically in this work, we evaluate higher-order, MU-MIMO smart-antenna technology added to the 802.11ac standard. MU-MIMO enables Ethernet gigabit speeds on wireless connections.

We have established the first efforts towards the successful and practical application of MU-MIMO to OFDM WLAN environments. We particularly tackle the signal processing, transmit mode selection, and rate selection components of MU-

MIMO in WLANs. For future work, we will analyze the impact of real WiFi channels on MU-MIMO performance. A sampling of other problems that we will tackle include low complexity channel feedback, user selection, and QoS issues. Finally, although we have discussed our findings in the context of 802.11 WLANs, our work is applicable to all next-generation OFDM MU-MIMO systems.

Chapter 8

Conclusions

The increasing complexity and availability of sophisticated wireless technology and systems over the past decade, has created ample opportunity and a need for wireless solution design today to evolve and adapt to address the challenges that emerge in complex systems. As portable devices increasingly rely on wireless connectivity, there is a need to develop solutions to handle the current and future traffic loads, and to deal with the unique problems and challenges resulting from this crowded, shared medium, such as congestion, QoE guarantees, hidden terminals, and interference. Human understanding of the means to best utilize the wireless technologies has not kept up with this technical evolution. We need to design automated and efficient solutions to modify and manage the attributes, parameters, and capabilities of wireless technologies for effective adoption.

My dissertation introduces several systems and resource management techniques that make significant contributions towards the design of such resource-efficient so-

lutions. Particularly, I take a step back and rethink wireless network design across all layers of the protocol stack, as it is clear that current usage of wireless spectrum has added new constraints, requirements, and expectations on wireless performance and availability. It is only through rethinking wireless network design can we come up with solutions that forego oudated assumptions, and move forward towards supporting, sustining, and enabling next-generation wireless systems and solutions. The following summary translates from chapters in this dissertation.

- I demonstrate the importance of designing application-aware mobile systems to improve performance on wireless networks and help meet QoE guarantees by making applications more bandwidth efficient. These systems are designed to deliver user-perceived application performance in mobile environments, where fading signal quality and intermittent connectivity are prevalent. Since applications have been designed with an assumption of constant and good connectivity, application performance typically suffers in such environments, and application operability is often compromised. I demonstrate the design of and need for effective wireless systems that adopt a top-down approach to address the challenges in mobile environments. I design a system that intelligently predicts the links that an application will request, or a client will access during usage sessions, and then goes and prefetches that information retroactively. This retroactive prefetching exploits every available

connection opportunity, thus maximizing the utility of connection periods in mobile environments. This system extends the online app experience to periods when the user is offline.

- I demonstrate the need to exploit the performance gains by capitalizing on the multiplicity of connection opportunities that are available on a single device today. With the widespread deployment of wireless networks, the benefits of such system design are clearly realizable to enhance user Internet experience. Using the characteristics of the wireless technologies, namely cost, delay, and data rate, I identify the cost factor that corresponds to different achieved performance data rates. These models serve as input to a system that can dynamically choose the best operating point based on a performance to cost tradeoff.

- I demonstrate a system design for the usage and sharing of white space spectrum. In wireless environments with unlicensed users, users can request and relinquish bandwidth at any time and in real-time. Based on predicted user behavior (i.e. cheating patterns), I infer features necessary in the design of an online spectrum distribution approach, and specifically a spectrum allocation approach through a spectrum auction framework. I develop a bidding model within that framework that discourages users or bidders from cheating and

ensuring that a spectrum auction that is built using this model will be truthful. Truthful spectrum auctions are necessary in order to ensure efficient use of spectrum bandwidth as well as social welfare between different bidders. Furthermore, I identify the features of a spectrum auction that can give the auctioneer the flexibility to select their tradeoff between spectrum efficiency and auctioneer revenue.

- I demonstrate that the benefits of wider channel widths can only be achieved through intelligent usage. Using the analysis from my measurements, I identify that there are more opportunities to exploit wider channel widths than initially projected. I infer that the performance of channel bonding not only relies on a link's current signal quality, but also on the strength of neighboring links and their physical rates. I develop a list of network characteristics in which wider channels can be exploited through channel bonding. Such findings serve as usage-terms (or model) for intelligently incorporating wider channel width, and specifically 40MHz channel operation in network deployments to maximize performance and efficiency. My work serves as a foundation on which channel management solutions for 802.11n networks can be built. I also believe that my work can be applied to the upcoming 802.11ac networks where 160MHz channels are supported.

- I demonstrate a method to exploit the added throughput gains provided by the smart-antenna Single-User Multiple-Input Multiple-Output (SU-MIMO or simply MIMO) technology in IEEE 802.11n WLANs. MIMO allows transmitters to increase the received SNR at a particular distance using *spatial diversity*. MIMO also allows transmitters to increase data rate using *spatial multiplexing*. The challenge in real networks is to identify when and how to exploit these added MIMO features. In my research, I therefore first demonstrate an understanding of the behavior and performance of MIMO through analysis of measurements performed in real testbed environments using 802.11n MIMO chipset cards. These findings allow me to develop an understanding and eventually motivate the design of a solution that exploits the benefits of MIMO. My analysis of MIMO measurements reveal that RSSI is an inaccurate link metric to characterize MIMO performance. I therefore identify a cost-effective and accurate link metric to reflect MIMO performance, and use this metric as a basis to design a rate adaptation solution that exploits MIMO features. Finally, I argue the need for closed-loop rate adaptation solutions in next-generation WiFi networks, to enable efficient use of complex PHY-layer features, namely MIMO. Through a testbed implementation of my closed-loop rate adaptation solution, I demonstrate the significant gains provided by this approach, compared to existing and leading solutions in literature.

- I demonstrate an intelligent approach to the application of Multi User (MU) MIMO (Multiple-Input Multiple-Output) in IEEE 802.11ac WLANs. MU-MIMO can provide significant spectral efficiency gains by allowing an AP to serve multiple clients simultaneously without requiring much added complexity or a larger form factor at the clients. Applying MU-MIMO to 802.11 WLANs, however, poses unique system constraints across the different layers of the protocol stack, from the physical layer signal processing and channel feedback to medium access and resource allocation. In my work, I have investigated and developed solutions towards the successful deployment of MU-MIMO systems in 802.11 environments. Specifically, I identify practical techniques for transmit precoding and receive signal processing, and formulate and address a critical OFDM MU-MIMO resource allocation and utilization problem, namely data rate selection.

Ultimately, I believe that through the design of efficient wireless systems, we can not only improve the state of current wireless networks, but also advance and impact a number of disciplines within and beyond computer science. The improvement in wireless systems has and will continue to push the frontier of what is possible using the wireless medium. In my research work, I have carried this vision and designed solutions that address emerging research challenges in modern wireless networks. Further, based on my findings and vision of the importance of designing resource-

efficient systems for emerging wireless networks, I believe the evolution of wireless system solutions should advance on two fronts to support innovations in mobile applications and the evolution of our interactions on the Internet in our daily lives.

- Unified application platform design for heterogeneous networks: Applications are not designed to exploit the opportunities and mitigate the challenges of multiple wireless technologies and their characteristics. As mobile and wireless technologies become more pervasive, it becomes increasingly crucial to address the impact of mobile environments on application performance and to design solutions that mitigate the negative impact of mobility on performance.

  In this dissertation, I have made the argument that transferring this responsibility to the lower layers is insufficient to address the specific QoE requirements of upper-layer applications. In fact, I demonstrate the need to design a unified platform over which different applications can be built to ensure effective usage of available networks in mobile environments as well as the resultant network performance variation. Further, with the wealth of information we are gathering from the end-user that allow us to profile user behavior, we need to develop mindful and intelligent methods of monetizing on this added information to provide a better user experience, and to push the edge of innovation and what we are capable of doing using the wireless medium.

- Model for future Internet access: Now that mobile operators are experiencing a capacity crunch due to the explosion in mobile data traffic growth, small cell towers have emerged as another potential solution to redistributing this traffic to alternative technologies and networks. However, we have yet to understand how to integrate small cell towers in the current design of wireless network deployments, systems, and protocols. Furthermore, as WiFi is projected to be integrated into virtually all 3G/4G small cells, the availability of wireless networks will flourish such that mobile devices can communicate with a combination of a macro-cell tower, small-cell tower, and WiFi access points. To exploit and accommodate this network multiplicity, we need to determine the methods to rectify and integrate these technologies together into a functional and economic wireless framework for Internet access.

  I believe we need to investigate the challenges and implications of assimilating the multiplicity of available wireless networks to exploit higher data rates and bandwidth availability. More specifically, it is important that we identify potential avenues and deployment scenarios for these multiple technologies such that they can coexist and provide improved and seamless user and application experiences.

Bibliography

[1] IEEE 802.11n: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 5: Enhancements for Higher Throughput. ANSI/IEEE Std 802.11n-2009, IEEE, Oct. 2009.

[2] IEEE 802.11ac-/D3.1 Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6GHz. IEEE-SA, August 2012.

[3] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *ACM SigComm*, Aug. 2004.

[4] N. Ahmed, U. Ismail, and S. Kashav. Online estimation of RF interference. In *CoNEXT*, 2008.

[5] A. Amini, D. Lu, and C. Edelman. Effects of high peak-to-average ratio on 5GHz WLAN power amplifier. In *DesignCon*, Feb. 2002.

[6] T. Armstrong, O. Trescases, C. Amza, and E. de Lara. Efficient and transparent dynamic content updates for mobile clients. In *ACM MobiSys*, 2006.

[7] M. Y. Arslan, K. Pelechrinis, I. Broustis, S. V. Krishnamurthy, S. Addepalli, and K. Papagiannaki. Auto-configuration of 802.11n WLANs. In *ACM CoNext*, Nov. 2010.

[8] E. Aryafar, N. Anand, T. Salonidis, and E. W. Knightly. Design and experimental evaluation of multi-user beamforming in wireless LANs. In *ACM MobiCom*, 2010.

[9] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. N. Levine, and J. Zahorjan. Interactive WiFi connectivity for moving vehicles. *Sigcomm Computer Communications Review*, 38(4):427–438, 2008.

[10] I. Broustis, K. Papagiannaki, S. Krishnamurthy, M. Faloutsos, and V. Mhatre. MDG: Measurement-driven guidelines for 802.11 WLAN design. In *ACM MobiCom*, June 2007.

[11] M. Buddhikot, G. Chandranmenon, S. Han, Y. W. Lee, S. Miller, and L. Salgarelli. Integration of 802.11 and third-generation wireless data networks. In *INFOCOM '03: Proceedings of Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, Mar. 2003.

[12] G. Caire, N. Jindal, M. Kobayashi, and N. Ravindran. Multiuser mimo achievable rates with downlink training and channel state feedback. *IEEE Transactions on Information Theory*, 56(6):2845–2866, June 2010.

[13] J. Camp and E. Knightly. Modulation rate adaptation in urban and vehicular environments: cross-layer implementation and experimental evaluation. In *ACM MobiCom*, Sept. 2008.

[14] E. Caron, P. K. Chouhan, and F. Desprez. Deadline scheduling with priority for client-server systems on the grid. In *IEEE/ACM Workshop on Grid Computing*, 2004.

[15] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl. A case for adapting channel width in wireless networks. In *ACM SigComm*, Aug. 2008.

[16] E. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[17] M. H. M. Costa. Writing on dirty paper (corresp.). *Information Theory, IEEE Transactions on*, 29(3):439–441, 1983.

[18] R. Crepaldi, J. Lee, R. Etkin, S.-J. Lee, and R. H. Kravets. CSI-SF: Estimating wireless channel state using CSI sampling and fusion. In *IEEE Infocom*, Orlando, FL, Mar. 2012.

[19] R. Crepaldi, J. Lee, R. H. Etkin, S.-J. Lee, and R. Kravets. Csi-sf: Estimating wireless channel state using csi sampling and fusion. In *IEEE Infocom*, March 2012.

[20] L. Deek, E. Garcia-Villegas, E. Belding, S.-J. Lee, and K. Almeroth. The impact of channel bonding on 802.11n network management. In *ACM CoNEXT*, Dec. 2011.

[21] L. Deek, E. Garcia-Villegas, E. Belding, S.-J. Lee, and K. Almeroth. Joint rate and channel width adaptation in 802.11 MIMO wireless networks. In *IEEE Secon*, June 2013.

[22] L. Deek, E. Garcia-Villegas, E. Belding, S.-J. Lee, and K. Almeroth. Joint rate and channel width adaptation in 802.11 MIMO wireless networks. In *IEEE Secon*, June 2013.
Nominee for the Best Paper Award.

[23] L. B. Deek, S. Thoubian, S. Jamijian, K. Harras, and H. Artail. Exploiting parallel networks in intermittently-connected mobile environments. In *WiMob '08: Proceedings of Fourth International Conference on Wireless and Mobile Computing, Networking and Communications*, Oct. 2008.

[24] K. Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Feb. 2003.

[25] F. Fietkau and D. Smithies. Linux wireless minstrel high throughput.

[26] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., 2003.

[27] M. Frodigh, S. Parkvall, C. Roobol, P. Johansson, and P. Larsson. Future-generation wireless networks. *IEEE Personal Communications*, 8(5):10–17, Oct. 2001.

[28] S. Gandhi et al. A general framework for wireless spectrum auctions. In *DySPAN*, 2007.

[29] E. Gelal, K. Pelechrinis, I. Broustis, S. Krishnamurhty, S. Mohammed, A. Chockalingam, and S. Kasera. On the impact of MIMO diversity on higher layer performance. In *IEEE ICDCS*, June 2010.

[30] M. Ghosh. A comparison of normalizations for zf precoded mu-mimo systems in multipath fading channels. *IEEE Wireless Communications*, 2(5):515–518, Oct. 2013.

[31] B. Ginzburg and A. Kesselman. Performance analysis of A-MPDU and A-MSDU aggregation in IEEE 802.11n. In *IEEE Sarnoff Symposium*, May 2007.

[32] A. Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[33] R. L. Graham et al. Optimization and approximation in deterministic sequencing and scheduling: A survey. In *Discrete Optimization II*, pages 287–326, 1979.

[34] I. P. V. Grigorios Tsoumakas, Ioannis Katakis. Effective and efficient multilabel classification in domains with large number of labels. In *ECML/PKDD Workshop on Mining Multidimensional Data*, 2008.

[35] T. Groves. Incentive in terms. *Econometrica*, 41:617–631, 1973.

[36] R. Gummadi and H. Balakrishnan. Wireless networks should spread spectrum based on demands. In *ACM Hotnets*, October 2008.

[37] M. T. Hajiaghayi, R. D. Kleinberg, and M. Mahdian. Online auctions with re-usable goods. In *Electronic Commerce*, 2005.

[38] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[39] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. 802.11 with multiple antennas for dummies. *ACM SigComm Computer Communications Review*, 40:19–25, January 2010.

[40] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *ACM Sigcomm*, Aug. 2010.

[41] K. Harras, M. Wittie, K. Almeroth, and E. Belding. Paranets: A parallel network architecture for challenged networks. In *HOTMOBILE '07: Proceedings of the Eighth IEEE Workshop on Mobile Computing Systems and Applications*, Nov. 2007.

[42] T. R. Henderson and Y. H. Katz. Transport protocols for internet-compatible satellite networks. *IEEE Journal on Selected Areas in Communications*, 17:326–344, 1999.

[43] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *IEEE Infocom*, April 2003.

[44] T. Hey, editor. *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons, Inc., 2003.

[45] B. D. Higgins, J. Flinn, T. J. Giuli, B. Noble, C. Peplin, and D. Watson. Informed mobile prefetching. In *ACM MobiSys*, 2012.

[46] C. Holman, K. A. Harras, K. C. Almeroth, and A. Lam. A proactive data bundling system for intermittent mobile connections. In *SECON '06: Proceedings of 3rd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, Sep. 2006.

[47] A. Houyou, R. Holzer, H. Meer, and M. Heindl. Performance of transport layer protocols in LEO pico-satellite constellations. Technical report, University of Passau, 2005.

[48] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *ACM MobiSys*, June 2012.

[49] J. Jia, Q. Zhang, Q. Zhang, and M. Liu. Revenue generation for truthful spectrum auction in dynamic spectrum access. In *MobiHoc*, 2009.

[50] A. Khan, V. Subbaraju, A. Misra, and S. Seshan. Mitigating the true cost of advertisement-supported 'free' mobile applications. In *HotMobile*, 2012.

[51] J.-K. Kim et al. Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment. *Journal of Parallel and Distributed Computing*, 67(2):154–169, 2007.

[52] S. E. Kim, J. A. Copeland, C. Chen, L. Wang, L. Liu, and W. Wu. Interworking between WLANs and 3G networks: TCP challenges. In *WCNC '04: Proceedings of 2004 IEEE Wireless Communications and Networking Conference*, Mar. 2004.

[53] W. Kim et al. An experimental evaluation of rate adaptation for multi-antenna systems. In *IEEE Infocom*, Apr. 2009.

[54] D. Kliazovich, F. Granelli, G. Pau, and M. Gerla. APOHN: subnetwork layering to improve TCP performance over heterogeneous paths. In *NGI '06: Proceedings of 2nd Conference on Next Generation Internet Design and Engineering*, Apr. 2006.

[55] E. Koukoumidis, D. Lymberpoulos, K. Strauss, J. Liu, and D. Burger. Pocket cloudlets. In *ASPLOS*, 2011.

[56] I. Koukoutsidis. TCP over 3G links problems and solutions. Technical report, Fourth - ICS, 2005.

[57] S. Lakshmanan, J. Lee, R. Etkin, S.-J. Lee, and R. Sivakumar. Realizing high performance multi-radio 802.11n wireless networks. In *IEEE SECON*, June 2011.

[58] S. Lakshmanan, S. Sanadhya, and R. Sivakumar. On link rate adaptation in 802.11n WLANs. In *IEEE Infocom mini-conference*, April 2011.

[59] R. Lavi, A. Mu'alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. In *Foundations of Computer Science*, 2003.

[60] R. Lavi and N. Nisan. Online ascending auctions for gradually expiring items. In *Symposium on Discrete Algorithms*, 2005.

[61] K. W. Lee, Y. B. Ko, and T. Nandagopal. Load mitigation in cellular data networks by peer data sharing over wlan channels. *Computer Networks*, 47(2):271–286, 2005.

[62] C.-Y. Li, C. Peng, S. Lu, and X. Wang. Energy-based rate adaptation for 802.11n. In *ACM Mobicom*, Aug. 2012.

[63] J. H. Lin, C. R. Dow, Y. H. Li, C. M. Lin, and T. C. Huang. An efficient channel allocation scheme in cell overlapping systems. In *CCNC '05: Proceedings of Second IEEE Conference on Consumer Communications and Networking*, Jan. 2005.

[64] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mobile Computer Communications Review*, 7(3):19–20, 2003.

[65] C. Liu and S. Baskiyar. Scheduling mixed tasks with deadlines in grids using bin packing. In *ICPADS*, 2008.

[66] D. Lymberopoulos, O. Riva, K. Strauss, A. Mittal, and A. Ntoulas. PocketWeb: instant web browsing for mobile devices. In *ASPLOS*, 2012.

[67] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Deroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition Journal*, Sept. 2012.

[68] M. Maheswaran et al. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In *IEEE Workshop on Heterogeneous Computing*, 1999.

[69] D. A. Menascé et al. Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures. *Journal of Parallel and Distributed Computing*, 28(1):1–18, 1995.

[70] R. Milkman. Broadband breakfast club: Fcc's mobile broadband agenda, Feb 22 2013.

[71] A. Mishra, V. Brik, S. Banerjee, A. Srinivasan, and W. Arbaugh. A client-driven approach for channel management in wireless LANs. In *IEEE Infocom*, April 2006.

[72] A. Mishra, V. Shrivastava, S. Banerjee, and W. Arbaugh. Partially overlapped channels not considered harmful. In *ACM SIGMETRICS*, June 2006.

[73] T. Moscibroda, R. Ch, Y. Wu, S. Sengupta, P. Bahl, and Y. Yuan. Load-aware spectrum distribution in wireless LANs. In *IEEE ICNP*, October 2008.

[74] S. Murphy. Mobile devices will outnumber people by the end of the year. Mashable, Feb. 6 2013.

[75] A. Naguib, A. Paulraj, and T. Kailath. Capacity improvement of base-station antenna arrays cellular cdma. In *Conference on Signals, Systems and Computers*, Nov. 1993.

[76] D. Nguyen and J. Garcia-Luna-Aceves. A practical approach to rate adaptation for multi-antenna systems. In *IEEE ICNP*, Oct. 2011.

[77] D. Niculescu. Interference map for 802.11 networks. In *IMC*, 2007.

[78] T. R. E. T. Noam Nisan and V. V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[79] C. Oestges and B. Clerckx. *MIMO Wireless Communications: From Real-World Propagation to Space-Time Code Design*. Academic Press, 2007.

[80] A. Parate, M. Böhmer, D. Chu, D. Ganesan, and B. M. Marlin. Practical prediction and prefetch for faster access to applications on mobile phones. In *UbiComp*, 2013.

[81] C. Partridge and T. J. Shepard. TCP/IP performance over satellite links. *IEEE Network*, 11(5):44–49, 1997.

[82] U. Paul, R. Crepaldi, J. Lee, S.-J. Lee, and R. Etkin. Characterizing WiFi link performance in open door networks. In *SECON*, June 2011.

[83] I. Pefkianakis, Y. Hu, S. H. Wong, H. Yang, and S. Lu. MIMO rate adaptation in 802.11n wireless networks. In *ACM MobiCom*, Sept. 2010.

[84] K. Pelechrinis, I. Broustis, T. Salonidis, S. V. Krishnamurthy, and P. Mohapatra. Design and deployment considerations for high performance MIMO testbeds. In *WICON*, November 2008.

[85] K. Pelechrinis, T. Salonidis, H. Lundgren, and N. Vaidya. Experimental characterization of 802.11n link quality at high rates. In *ACM WiNTECH*, September 2010.

[86] F. Peng, J. Zhang, and W. E. Ryan. Adaptive modulation and coding for IEEE 802.11n. In *IEEE WCNC*, pages 656–661, March 2007.

[87] S. Pollin and A. Bahai. Performance analysis of double-channel 802.11n contending with single-channel 802.11. In *IEEE International Conference on Communications (ICC)*, june 2009.

[88] J. S. R. Gass and C. Diot. Measurements of in-motion 802.11 networking. In *WMCSA '06: In Proceedings of Seventh IEEE Workshop on Mobile Computing Systems and Applications*, Apr. 2006.

[89] H. Rahul, F. Edalat, D. Katabi, and C. G. Sodini. Frequency-aware rate adaptation and MAC protocols. In *ACM MobiCom*, Sept. 2009.

[90] N. Ravindran. *MIMO wireless communications with limited feedback*. PhD thesis, University of Minnesota, 2012.

[91] N. Ravindran and N. Jindal. Limited feedback-based block diagonalization for the MIMO broadcast channel. *IEEE Journal on Selected Areas in Communications*, 26(8):1473–1482, 2008.

[92] N. Ravindran and N. Jindal. Multi-user diversity vs. accurate channel state information in MIMO downlink channels. *IEEE Transactions on Wireless Communications*, 11(9):3037–3046, Sept. 2012.

[93] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Measurement-based models of delivery and interference in static wireless networks. In *ACM SigComm*, Sept. 2006.

[94] M. Sadek, A. Tarighat, and A. Sayed. A leakage-based precoding scheme for downlink multi-user MIMO channels. *IEEE Transactions on Wireless Communications*, 6(5):1711–1721, May 2007.

[95] M. Satyanarayanan, J. Kistler, and E. Siegel. Coda: A resilient distributed file system. In *IEEE Workshop on Workstation Operating Systems*, 1987.

[96] W.-L. Shen, Y.-C. Tung, K.-C. Lee, K. C.-J. Lin, S. Gollakota, D. Katabi, and M.-S. Chen. Rate adaptation for 802.11 multiuser MIMO networks. In *ACM MobiCom*, 2012.

[97] C. Shepard, N. Anand, and L. Zhong. Practical performance of MU-MIMO precoding in many-antenna base stations. In *ACM Workshop on Cellular Networks (CellNet)*, June 2013.

[98] D.-S. Shiu, G. Foschini, M. Gans, and J. Kahn. Fading correlation and its effect on the capacity of multielement antenna systems. *IEEE Transactions on Communications*, 48(3):502–513, Mar. 2000.

[99] V. Shrivastava, S. Rayanchu, J. Yoonj, and S. Banerjee. 802.11n under the microscope. In *ACM IMC*, Oct. 2008.

[100] Q. Spencer, A. Swindlehurst, and M. Haardt. Zero-forcing methods for downlink spatial multiplexing in multiuser MIMO channels. *IEEE Transactions on Signal Processing*, 52(2):461–471, February 2004.

[101] S. Subramanian, S. Sivakumar, W. J. Phillips, and W. Robertson. Investigating TCP performance issues in satellite networks. In *CNSR '05: In Proceedings of the 3rd Annual Conference on Communicatio Networks and Services Research*, May 2005.

[102] A. Takefusa et al. A study of deadline scheduling for client-server systems on the computational grid. In *High Performance Distributed Computing*, 2001.

[103] Texas Instruments. WLAN channel bonding: Causing greater problems than it solves. Technical report, September 2003.

[104] D. Tse and P. Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[105] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *Data Warehousing and Mining*, 2007.

[106] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12, 2011.

[107] W. Vickery. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[108] V. Visoottiviseth, T. Piroonsith, and S. Siwamogsatham. An empirical study on achievable throughputs of IEEE 802.11n devices. In *IEEE WiOPT*, June 2009.

[109] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. In *ACM SigComm*, Aug. 2009.

[110] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. *SIGOPS Operating System Review*, 36(SI):255–270, 2002.

[111] M. P. Wittie, B. Stone-Gross, K. C. Almeroth, and E. M. Belding. MIST: Cellular data network measurement for mobile applications. In *BROADNETS '07: In Proceedings of Fourth International Conference on Broadband Communications, Networks and Systems*, Sep. 2007.

[112] M. Wong, J. M. Gilbert, and C. H. Barratt. *Wireless LAN using RSSI and BER parameters for transmission rate adaptation*. US patent 7,369,510, 2008.

[113] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *ACM MobiCom*, Sept. 2006.

[114] E. Wyatt. Fcc backs proposal to realign airwaves. The NY Times, Sept. 28 2012.

[115] W. H. Xi, A. Munro, and M. Barton. Link adaptation algorithm for the IEEE 802.11n MIMO. In *Networking LNCS*, 2008.

[116] P. Xu and X.-Y. Li. Online market driven spectrum scheduling and auction. In *ACM Workshop on Cognitive Radio Networks*, 2009.

[117] P. Xu, S. Wang, and X.-Y. Li. SALSA: Strategyproof online spectrum admissions for wireless networks. *IEEE Transactions on Computers*, 99(PP):1–1, 2010.

[118] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman. Identifying diverse usage behaviors of smartphone apps. In *IMC*, 2011.

[119] T. Yan, D. Chu, D. Ganesan, A. Kansal, and J. Liu. Fast app launching for mobile devices using predictive user context. In *ACM MobiSys*, 2012.

[120] H. Yang. A road to future broadband wireless access: MIMO-OFDM-based air interface. *IEEE Communications Magazine*, 43(1):53–60, Jan. 2005.

[121] J. Yang. An efficient fault-tolerant distributed channel allocation algorithm for cellular networks. *IEEE Transactions on Mobile Computing*, 4(6):578–587, 2005.

[122] T. Yoo and A. Goldsmith. Optimality of zero-forcing beamforming with multiuser diversity. In *IEEE ICC*, May 2005.

[123] T. Yoo and A. Goldsmith. On the optimality of multiantenna broadcast scheduling using zero-forcing beamforming. *IEEE Journal on Selected Areas in Communications*, 24(3):528–541, March 2006.

[124] X. Zhou et al. eBay in the sky: Strategy-proof wireless spectrum auctions. In *MobiCom*, 2008.

[125] X. Zhou and H. Zheng. TRUST: A general framework for truthful double spectrum auctions. In *INFOCOM*, 2009.

[126] X. Zhou and H. Zheng. Breaking bidder collusion in large-scale spectrum auctions. In *MobiHoc*, 2010.